НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

«До захисту допущено»
Завідувач кафедри
_____ С. Г., Стіренко
«___»_____20__ р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6. 050102 – «Комп'ютерна інженерія»

на тему: «*Система забезпечення якості освітньо-наукової діяльності аспірантури*»

Виконав:
Студент IV курсу, групи ІО-64
Жинжер Олександр Сергійович _____

Керівник:
доц. каф. ОТ
к.т.н., доц. Ткаченко В.В. _____

Консультант з нормоконтролю:
проф. каф. ОТ, д.т.н., проф.
Сімоненко В.П. _____

Рецензент:
доц. каф АУТС, к.т.н., доц.
Сперкач Майя _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський) Спеціальність – 123 «Комп'ютерна інженерія» Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Сергій СТІРЕНКО
«___»_____2020 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Жинжеру Олександру Сергійовичу

1. Тема проєкту «*Система забезпечення якості освітньо-наукової діяльності аспірантури*», керівник проєкту Ткаченко Валентина Василівна, старший викладач, затверджені наказом по університету від «07» травня 2020 р. № 1081-с
2. Термін подання студентом проєкту *26 травня 2020р.*
3. Вихідні дані до проєкту *див. технічне завдання*
4. Зміст пояснювальної записки *Аналіз і характеристика об'єкта проектування, обґрунтування оптимального варіанта реалізації мети цієї роботи, розробка додатку: вибір технологій та їх обґрунтування, основні рішення з реалізації додатку. Висновки*
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) *приниципова схема, функціональна схема, структурна схема.*

6. Консультанти розділів проєкту.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В. П., професор, д.т.н.		

7. Дата видачі завдання 01.09.2019

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Затвердження теми роботи	01.09.2019	
2.	Вивчення та аналіз завдання	15.12.2019-15.03.2020	
3.	Розробка архітектури додатку	15.03.2020-25.03.2020	
4.	Написання програмної частини	25.03.2020-05.04.2020	
5.	Тестування та виправлення помилок	05.04.2020-15.04.2020	
6.	Оформлення пояснювальної записки	15.04.2020-20.05.2020	
7.	Захист програмного продукту	25.04.2020	
8.	Передзахист	26.05.2020	
9.	Захист	18.06.2020	

Студент Олександр ЖИНЖЕР

Керівник Валентина ТКАЧЕНКО

АНОТАЦІЯ

В даній дипломній роботі було створено систему забезпечення якості освітньо-наукової діяльності аспірантури. У ході роботи були досліджені можливості платформи у порівнянні з її конкурентами та була проаналізована можливість оптимізації обробки користувацьких запитів в контексті платформи. У якості серверної частини були використані API платформи. А для написання користувацьких інтерфейсів було використано бібліотеку React.

АННОТАЦИЯ

В данной дипломной работе была создана система обеспечения качества образовательно-научной деятельности аспирантуры. В ходе работы были исследованы возможности платформы по сравнению с ее конкурентами и была проанализирована возможность оптимизации обработки пользовательских запросов в контексте платформы. Вкачестве серверн̆ части были использованы API платформы. А для написания пользовательских интерфейсов было использовано библиотеку React.

ABSTRACT

In this thesis, a system to ensure the quality of educational and scientific activities of graduate school is created. In the process of the work, the possibilities of the platform in comparison with competitors were investigated and the possibility of optimizing the processing of user requests in the context of the platform was analyzed. Platform APIs were used as the server part. And the React library was used to write user interface.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	6408.02.000 ТЗ	Технічне завдання	3	
3	А4	6408.03.000 ПЗ	Пояснювальна записка	70	
4	А4	6408.04.000 Д1	Принципова схема	1	
5	А4	6408.05.000 Д2	Функціональна схема	1	
6	А4	6408.06.000 Д3	Структурна схема	1	

					ДП.6408.01.000 ВП		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Жинжер О.С.			*Система забезпечення якості освітньо-наукової діяльності аспірантури Пояснювальна записка*	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Ткаченко В.В.					
Реценз.						НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІО-64	
Н. Контр.		Сімоненко В. П.					
Затвердив							

Технічне завдання

до дипломного проєкту

На тему “Система забезпечення якості освітньо-наукової діяльності аспірантури”

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: “Система забезпечення якості освітньо-наукової діяльності аспірантури”. Область застосування: програма може використовуватися в вищих навчальних закладах.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського дипломного проекту, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки є створення автоматизовананої системи забезпечення якості освітньо-наукової діяльності аспірантури.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література, публікації в спеціалізованих періодичних виданнях, довідники по платформах дистанційного навчання, публікації в мережі Інтернет по даній темі.

5. ТЕХНІЧНІ ВИМОГИ

Додаток, що розробляється повинен бути зручним у використанні. Повинен мати інтерфейс з мінімальним навантаженням, для швидної орієнтації. Мати можливість для швидного пошуку потрібних аспірантів. Мати можливість додавати нових аспірантів. Повинен працювати так само швидко або швидше аніж схожі вбудовані додатки.

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення необхідної літератури	19.02.2020
Складання і узгодження технічного завдання	06.03.2020
Написання вступної частини та огляд рішень	19.03.2020
Розробка архітектури додатку	03.04.2020
Написання програмної частини	10.04.2020
Тестування та виправлення помилок	01.05.2020
Оформлення документації дипломного проекту	15.05.2020
Попередній захист та проходження нормативного контролю	29.05.2020
Захист дипломного проекту	18.06.2020

Пояснювальна записка
до дипломного проєкту

На тему “Система забезпечення якості освітньо-наукової діяльності аспірантури”

ЗМІСТ

ВСТУП

В умовах розбудови української держави, глибоких і динамічних перетворень, що відбуваються в усіх сферах нашого суспільства, суттєвих змін зазнає і система освіти. Стратегію змін у системі професійної освіти в Україні чітко викладено у «Національній стратегії розвитку освіти в Україні у 2012-2021 рр.» щодо впровадження комп‘ютерних технологій навчання та підвищення якості професійної підготовки майбутніх фахівців. Підготовка спеціалістів із вищою освітою вимагає підвищення їхнього загального рівня інформатизації, оволодіння ними інформаційнокомунікаційними технологіями та сучасними методами навчання, засобами комп‘ютерних технологій, засобами інтелектуальних та алгоритмічних комп‘ютерних технологій у навчальному процесі.

У навчальних закладах створюються системи з базами даних викладачів, студентів та інших співробітників ВНЗ(Вищий Навчальний Заклад), автоматизується процесс вступної компаніїї та зарахування студентів, надається електронний розклад, контроль академічної успішності.

І це не дивно, адже автоматизація всіх цих процессів надеє можливість зекономити час та ресурси при вирішенні складних управлінських завдань. Вона виключає людський фактор і дозволяє контролювати виконання трудових функцій всієї установи. В результаті автоматизації ВНЗ отримує безліч переваг, таких як:

1. Підвищення наукового потенціалу вузу.
2. Елктронний документо обіг.
3. Скорочення витрат на канцелярію та зберігання всіх документів в архівах.
4. Більша привабливість для абітуріентів.

Таким чином, інформатизації освіти в сучасному світі відводиться дуже суттєва роль, оскільки саме цей процес є «двигуном» майбутнього, саме від цього процесу залежить успіх якості освіти країни, її технічний потенціал, а безпосередньо успіх даного процесу буде залежати від висококласних фахівців, здатних усунути насущні проблеми ВНЗ і просунути його на новий, більш високий рівень.

Метою данної роботи є створення системи ефективного управління аспірантурою в ВНЗ, та автоматизація таких процесів як зарахування на аспірантуру, переведення на наступний курс, відрахування, зберігання та представлення даних про аспірантів.

РОЗДІЛ 1
ОГЛЯД РІШЕНЬ СИСТЕМ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ОСВІТНЬО-НАУКОВОЇ ДІЯЛЬНОСТІ

1.1. Загальні положення

З початоком XXIX ст. відбувається швидкий розвиток інформації технологій та їх швидке проникнення у всі сфери суспільного життя, зокрема у галузі освіти, створюючи передумови для переходу країни до більшого рівень розвитку. Один із напрямків сучасної державної політики України в освіті полягає в удосконаленні інфраструктури інформаційного освітнього простору це відображено в Законі України “Про Національну програму інформатизації”[1]. Важливим напрямком інформатизації Понятійний апарат інформатизації відображено в Законі України “Про національну програму інформатизації”[2]. Визначається, що інформатизація – сукупність взаємопов'язаних організаційних, правових, політичних, соціальноекономічних, науково-технічних, виробничих процесів, що спрямовані на створення умов для задоволення інформаційних потреб громадян та суспільства на основі створення, розвитку і використання інформаційних систем, мереж, ресурсів та інформаційних технологій, які побудовані на основі застосування сучасної обчислювальної та комунікаційної техніки. Запровадження ІКТ(Інформаційно комунікаційні технології) у сферу освіти є елементом державної політики. У Законі чітко визначені функції органів державної влади у реалізації Національної програми інформатизації. Вони повинні в межах їх компетенції здійснювати такі функції у процесі інформатизації:

1. Захист авторського права на бази даних і програми, створені для потреб інформатизації та особистої інформації.
2. Встановлення стандартів, норм і правил використання засобів інформатизації.
3. Забезпечення доступу громадян та їх об'єднань до інформації органів державної влади та органів місцевого самоврядування, а також до інших джерел інформації.

4. Визначення пріоритетних напрямів інформатизації з метою подальшої її підтримки шляхом державного фінансування та пільгового оподаткування.
5. Інформатизацію науки, освіти, культури, охорони довкілля та здоров'я людини, державного управління, національної безпеки та оборони держави, пріоритетних галузей економіки.

Найбільша проблема в Україні - це створення освітніх інформаційних ресурсів. Як справедливо зазначає В. Лунячек, в Україні реалізується друга програма інформатизації сфери освіти, але створення навчальних місць для більшої кількості відділів (кабінетів) освіти та більшості навчальних закладів відбувається дуже повільно. Закон України “Про основні засади розвитку інформаційного суспільства в Україні за період 2007-2015 років”[3] визначає стратегічні цілі розвитку інформаційного суспільства в Україні:

1. Забезпечення комп'ютерної та інформаційної грамотності населення, насамперед шляхом створення системи освіти, орієнтованої на новітні ІКТ, розвинені особистості.
2. Створення національних інформаційних систем, особливо у галузі охорони здоров'я, освіти, науки, культури, охорони навколишнього середовища.
3. Використання ІКТ для поліпшення державного управління, відносин між державою та громадянами, створення електронних форм взаємодії між органами державної влади та місцевого самоврядування та фізичними та юридичними особами.

Серед основних напрямків розвитку інформаційного суспільства в Україні:

1. Забезпечення вільного доступу населення до телекомунікаційних послуг, зокрема до Інтернету, ІКТ та інформаційних ресурсів.
2. Дати можливість кожному набувати знань, умінь та навичок завдяки використанню ІКТ в освіті, вихованні та освіті.

3. Створити умови для забезпечення комп’ютерної та інформаційної грамотності всіх верств населення, встановлення системи мотивацій щодо впровадження та використання ІКТ для створення широкого попиту на такі технології в усіх сферах суспільства.

У той же час, з технологічним аспектом комп’ютеризації університетів, реалізація програми неможлива без розвиненої системи інформаційних послуг, які вони надають. Інформаційні сайти університетів відіграють важливу роль у цьому процесі. Вони повинні стати не лише джерелами інформації про діяльність установи, а й засобом комунікації та інтеграції в інформаційний простір.

Останнім часом значна увага приділяється модернізації соціальної та економічної сфер. Це не лише питання розробки певних механізмів взаємодії та реалізації низки державних програм, а ефекту, пов'язаного з їх реалізацією. Орієнтація на економіку знань в цьому плані викликає багато різних питань у галузі освіти, оскільки ця сфера є основною “кузнею кадрів” нового суспільства.

На сьогодні при оцінці якості освіти слід виділити низку негативних проявів:

1. Низька кваліфікація значної частини адміністративно-управлінського персоналу не дозволяє розвивати систему освіти на основі впровадження інформаційних систем.
2. Погана сприйнятливість традиційної системи освіти до нових підходів.
3. Брак кваліфікованого персоналу.
4. Недостатньо розроблені механізми залучення громадських та професійних організацій до вирішення питань формування та реалізації освітньої політики.
5. Немає умов для розробки незалежних форм оцінювання якості освіти, а також механізмів виявлення, підтримки та поширення кращих прикладів інноваційної освітньої діяльності.

З іншого боку, не можна не помітити позитивних тенденцій, які проявляються у вигляді реалізації пріоритетних загальнодержавних проектів, різних національних та регіональних програм підтримки, а також всебічної

логістичної модернізації освіти. Останнє - це процес, який, на мою думку, повиннен базуватися не лише на минулому світовому та вітчизняному досвіді, а й на прогнозуванні загальних тенденцій у майбутньому, необхідно не лише прийняти та адаптувати сучасні технології в освіті, а й створити на існуючій базі якісно нові навчальні заклади, з активними державними та приватними інвестуванням.

У сучасному суспільстві набувається здобуття нових знань, розробка нових технологій, методів управління соціальними та науковими процесами. Будь-який вид діяльності повинен пройти певні етапи, які безпосередньо пов'язані зі збором інформації, її аналізом, підбором пріоритетних завдань, пошуку оптимальних рішень проблем, формування підходу для досягнення поставлених цілей. Сьогодні студенти та викладачі університету є членами соціального середовища, в якому існує величезний потік інформації, який постійно оновлюється, і через обмежені можливості не можна повною мірою скористатися цим “величезним потоком”. У нинішніх умовах людство підійшло до такого процесу, як “комп'ютеризація”. Процес інформатизації нерозривно пов'язаний зі статусом сучасного суспільства, тобто зі статусом інформаційного суспільства, в якому переважає інформація, її якість, свобода, прозорість та доступність.

Інформатизація - це масштабний процес, який зачіпає всі сфери суспільного життя, спрямовані на задоволення інформаційних потреб людей, а також на побудову потужної телекомунікаційної інфраструктури.

Одна з основних позицій в інформатизації суспільства відводиться інформатизації в освіті. Інформатизація освіти - це процес забезпечення освітньої системи теорією та практикою розвитку та використання нових інформаційних технологій, спрямованих на досягнення цілей викладання певного університету.

Процес інформатизації освіти включає в себе систему заходів:

1. Оснащення закладів освіти та органів управління освітою апаратними та програмними засобами інформаційних технологій.

2. Підключення по високошвидкісних каналах до регіональних, національних і міжнародних комп'ютерним освітніх мереж, до глобальної мережі Інтернет.
3. Створення і розміщення в мережі Інтернет інформаційних ресурсів освітнього призначення, інтеграція різних баз даних на регіональному та державному рівні.
4. Формування інформаційної культури у всіх учасників освітнього процесу: співробітників, викладачів, студентів.
5. Створення системи безперервного навчання викладачів новим інформаційним технологіям (курси, експрес-курси, міні семінари, постійно-діючі семінари, конференції.).

Процес інформатизації освіти має на увазі не тільки застосування в школах, університетах та інших навчальних закладах новітніх інформаційних технологій, а й вдосконалення системи організації навчальної діяльності шляхом трансформації методів і форм піднесення інформації з метою пробудження в студентів інтересу до отримання нових знань, розвитку у них творчої активності. Безсумнівно, в першу чергу необхідним є створення інформаційно-технологічного середовища з використанням новітніх проекторів, екранів, комп'ютерів та іншої техніки, що дозволяє розкрити інформацію через презентації, конференції, семінари.

Тобто процес інформатизації освіти неможливий без участі кваліфікованого фахівця який глибоко знає процеси, що відбуваються в освіті, вміє використовувати ІТ(Інформаційні технології) у своїй професійній сфері. Однак інформатизація в освіті має ряд суперечностей:

1. Більшість викладачів у своїх дисциплінах стикаються з неякісним програмним забезпеченням, з технічними труднощами в організації навчального процесу, а тому відмовляються від використання комп’ютерних технологій на уроках та в управлінні університетом.
2. Більшість університетів не забезпечені повністю для освєння всього програмного пакету.

Тому що одним із основних напрямків інформатизації є будівництвоінформаційного освітнього простору, в цьому контексті виникають такі проблеми, як:

1. Відсутність єдиного програмного стандарту.
2. Брак технічного персоналу для обслуговування технічних пристроїв.
3. Відсутність єдиної інформаційної культури.
4. Більшість сільських шкіл та обласних університетів погано обладнані, що значно ускладнює навчальний процес.
5. Швидке старіння курсу, відсутність можливостей для змін або доповнення існуючого курсу новими модулями.
6. Проблеми, пов'язані зі створенням специфікацій для розвинених закладів та подальшою стандартизацією освітніх технологій інформатизації.

Використання інформаційних технологій в університетах дозволяє вирішити наступні проблеми:

1. Зменшити навантаження на викладачів та керівництво.
2. Розвиток умінь і навичок вирішувати типові практичні завдання.
3. Розробка системи автоматичного аналізу ефективності учнів.
4. Поліпшення спілкування між студентами, викладачами та керівництвом.
5. Підтримувати та розробляти європейські стандарти якості освіти.
6. Зняти бар'єри для підвищення мобільності студентів, викладачів та дослідників.
7. Запровадити сучасні підходи до інтеграції вищої освіти та науки у підготовку магістрів та аспірантів.
8. Забезпечити подальший розвиток самостійності та самоврядування в системі вищої освіти та науки.
9. Інформатизація освіти, яка носить системний характер.
10. Індивідуалізація вищої освіти з метою розвитку здібностей, схильностей та інтересів студентів.

1.2. Методи підвищення рівня інформатизації ВНЗ України

Рекомендується найближчим часом значно підвищити взаємну обізнаність розробників та користувачів за рахунок розвитку та використання інформаційно-комунікаційних технологій. Поширюючи позитивний досвід деяких університетів у галузі інформатизації освіти, слід мати на увазі, що розвиток у напрямку більшого використання інформаційних технологій у вищій освіті в деяких випадках повільний - важко реалізувати історично міцну мережу традиційних взаємопов'язаних стереотипів, моделей та взаємозв'язків. Тому інформатизація навчального процесу повинна починатися з наступних етапів:

1. Розвиток логістики комп'ютерного обладнання.
2. Створення комп'ютерних систем для управління університетською діяльністю.
3. Використання сучасних інформаційних технологій у навчальному процесі.
4. Розробка інтерактивних комплексів навчально-методичного забезпечення навчальної дисципліни.

Бюджет, технічні обмеження повинні бути ретельно враховані при розробці та реалізації спеціальних програм інформатизації вищої освіти. Цілі та завдання інформатизації вищої школи. У XXIX столітті вища освіта стала джерелом стратегічних ресурсів - людського капіталу та знань, що визначають людські та технологічні ресурси суспільства. Так що держава, яка прагне зайняти гідне місце у міжнародному співтоваристві, з будь-якої причини не може відмовитись прийняти певну стратегію своєї поведінки стосовно освіти.

Для вирішення вищезазначених проблем в першу чергу необхідно реформувати систему освіти шляхом покращення якості освіти без збільшення кількості. Україна повинна ретельно переглянути методики та розробки в галузі освіти, запропоновані Європою та США. Мета інформатизації освіти - це глобальна раціоналізація інтелектуальної діяльності за допомогою використання нових інформаційно-комунікаційних технологій, докорінне підвищення ефективності та

якості навчання до рівня, досягнутого в розвинених країнах, тобто. Тренування з новим видом мислення, що задовольняє сучасне суспільство. У результаті досягнення цієї мети суспільство повинно забезпечити масову комп’ютерну грамотність та формування нової інформаційної культури мислення шляхом індивідуалізації освіти.

1.3. Приклади типових інфраструктур для систем управління ВНЗ

Територіально більшість університетів мають структуру, розподілену в певному регіоні, яка може бути з'єднана на мережевому рівні шляхом створення приватної регіональної комп'ютерної мережі, ядром якої є корпоративна комп'ютерна мережа основного кампусу. Так, мережа Державного університету економіки та послуг у Владивостоці складається з корпоративної мережі головного кампусу та місцевих відділень у містах. Елементи регіональної магістральної мережі, що базується на маршрутизації CISCO[4] та обладнання Natek[5], використовуються для інтеграції галузевих мереж та основної мережі. Ядро корпоративної мережі захищено апаратним брандмауером FireBok 2500[6].

Архітектура регіональної мережі може бути представлена на декількох рівнях:

1. Рівень мережі підтримки філії.
2. Рівень вузлів доступу.
3. Рівень доступу до абонентських мереж.
4. Рівень інтеграції та взаємодії із зовнішніми мережами (мережі загальнодоступних даних, мережі Інтернет-провайдерів, відомчі мережі).
5. Рівень корпоративної мережі.
6. Рівень управління мережею.

Мережа підтримки філій у Владивостоці побудована на основі волокон з оптичними волокнами, що з'єднують вузли доступу (змінного струму). UD[7] виконує такі функції:

1. Підключення абонентських мереж доступу.

2. Підключення до зовнішніх мереж (особливо, підключення до Інтернет-провайдерів).
3. Вхідні, вихідні та комутаційні потоки даних опорної мережі.

Устаткування для мультиплексування UD працює за технологіями синхронної цифрової ієрархії. Надалі можна перейти до повної інтеграції всіх сервісів у загальну мережу банкоматів. В даний час переважна більшість постачальників послуг телекомунікацій використовує технологію SDH[8] для створення домашніх мереж. Цей вибір визначається надійністю мережі, низьким часом відновлення, прозорістю передачі різних видів трафіку, простотою обслуговування та можливістю збільшення пропускної здатності, простотою управління. SDH - це основна технологія створення первинних мереж для зв'язку та передачі даних. Мережі голосу, та банкоматів будуються поверх SDH. Основний потік компонентів у мережі SDH - це потік (2 Мбіт / с). Мультиплексори комбінують складові потоки в сукупні потоки для передачі по оптичним або електричним каналам. Перший рівень в ієрархії сукупних потоків. Загальна ємність становить 155 Мбіт / с, по цьому каналу можуть передаватися 63 потоки.

Пристрій ієрархії SDH такий, що мультиплексор може витягувати з сукупного потоку будь-який даний компонентний потік без повного демультиплексування всього каналу. Відповідно, можна вставити потік компонентів у сукупний потік. Мультиплексори, які виконують такі функції, називаються селективними множинними добавками (або введення-виведення, доповненнями мультиплексорів). Магістральне рішення на основі SDH має такі основні характеристики.

Рішення про корпоративне рішення щодо вибору однієї або декількох операційних систем для використання в університеті виникає через багато обставин. Перш за все (ми обговорюємо варіант легального використання комерційної продукції) враховуйте економічні міркування та досвід IT-персоналу. Відомі приклади університетів, які приймають подібні рішення, проте кілька програм підтримки Microsoft, зокрема, програмне забезпечення, дозволяють

юридично та розумно використовувати програмне забезпечення компанії для створення інформаційної інфраструктури. У вузли продуктів Microsoft[9] в основному використовуються на сервері та клієнтах, а FreeBSD[10], встановлюються як додаткові дані на корпоративних серверах.

Для забезпечення контролю та управління доступом до ресурсів та послуг СНД необхідний спеціалізований сервер для управління обліковими записами користувачів. Можна скористатися службою каталогів, яка є розподіленим сховищем даних, уніфікований доступ якого здійснюється за допомогою протоколу. Служба каталогів підтримує ієрархічну структуру даних, яка дозволяє структурувати інформацію, наприклад, відповідно до організаційної структури університету.

На ринку програмного забезпечення відомо кілька продуктів: LDAP Sun-сервер, LDAP-сервер[11] від Oracle[12]. Порівняльний аналіз показав, що кожна з цих систем має переваги та недоліки. Суттєвою перевагою є спрямованість не тільки на зберігання та авторизацію користувачів, але й на централізоване управління робочими станціями та серверами корпоративної мережі. Використання таких інструментів, як кафедри, групова політика та дозволяє адміністраторам ефективно керувати користувачами, мережевими ресурсами, комп'ютерами в межах всієї корпоративної мережі університету та кількістю користувачів, у яких може бути десятки тисяч. Ще одна перевага пов'язана з можливістю інтеграції в інші продукти, що покращує уніфікацію доступу до ресурсів університету СНД. Враховуючи ці переваги, вирішив використовувати технологію Active Directory.

При проектуванні дерева доменів вищої освіти в структурі можна керуватися такими критеріями:

1. Домени можуть бути організовані за атрибутом ролі (кореневий домен, домен службовців, домен учнів, адміністративний сервер).
2. Домени можуть бути організовані на територіальній основі (домени окремих структурних одиниць, які територіально відокремлені).

3. Змішане рішення, коли поділ у головному кампусі ґрунтується на ролі, а домени віддалених гілок об’єднуються відповідно до територіальних.

Третій варіант обраний у. В управлінні університетом існує кілька бізнес-процесів, які слід автоматично підтримувати на рівні контролю доступу до інформаційних ресурсів, включаючи:

1. Найм та звільнення працівника.
2. Зарахування (відрахування) студента до університету.
3. Переведення (або зміна посади) працівника в університеті.
4. Переведення студента на іншу спеціальність (другий факультет, другий курс тощо).
5. Зміна організаційної структури університету (злиття / передача / вилучення / створення / підпорядкування підрозділів).

Специфіка сучасного стану застосування інформаційних технологій на корпоративному рівні полягає в втраті інтересу до традиційної моделі підвищення функціональності додатків за рахунок придбання багатьох програмних продуктів або застосування системних пакетів. Рішення для інтеграції програм на основі архітектури порталу, асинхронних веб-сервісів та Enterprise Service Bus[13] сьогодні набагато затребувані. З розвитком цієї архітектури реалізація певних функцій буде виконуватися не корпоративними додатками, а інформаційними компонентами або веб-сервісами, тому завдання формування СНД є надзвичайно важливим.

Для створення СНД третього рівня можна використовувати декілька організаційних та технологічних підходів. Найпоширенішими є дві - засновані на концепції використання монолітних систем класу ERP(Enterprise Resource Planning — Система планування ресурсів) та засновані на розробці окремих додатків з подальшою їх інтеграцією, використовуючи концепцію Веб-сервісів (Веб-сервісів). Недоліки використання ERP-систем обговорювалися, наприклад, у літературі. Можна додати, що додатковою перешкодою у застосуванні ERP-систем у університеті, крім загальних, притаманних будь-якій реалізації ERP-рішень, є

зазначена вище специфіка університету як предмета інформатизації. Ця специфіка значно ускладнює повну реалізацію системи ERP в університеті. Однак використання ERP-системи може бути хорошим вибором, якщо інтеграція ERP-системи в єдине інформаційне середовище університету.

Підхід, заснований на консолідації інформаційних служб за допомогою веб-служб, дозволяє консолідувати доступ до даних та програм. Такий підхід, з одного боку, забезпечить перенесення в сучасне інформаційне середовище застарілих додатків та подальше використання наявних даних, а з іншого - надалі впровадить нові інформаційні послуги на основі єдиної технологічної політики, що спрощує підтримку та розвиток корпоративного інформаційного середовища.

Ще один привабливий підхід, заснований на веб-сервісах, полягає в тому, що університети зазвичай вже мають функціонуючі інформаційні системи, функціональність яких цілком адекватна стратегічним та поточним цілям розвитку університету. Набагато ймовірніше, що IT-проект буде успішним, що не потребує докорінної зміни всіх раніше розроблених та використовуваних інформаційних систем, а лише певної їх модернізації для інтеграції в єдине середовище.

По-друге, характеристики інноваційного розвитку в конкретному університеті та інформатизація, безумовно, підпадають під цю категорію, часто вимагаючи швидкої зміни або підвищення функціональності певної IT-системи, що значно полегшує впровадження власного персоналу на основі компонентних архітектур та веб-сервісів.

Додатковим голосом для використання веб-служб є просторовий розподіл університету. Інтегрування даних із структур, розташованих на великій відстані одна від одної, може бути найбільш ефективно виконано за допомогою веб-служб.

Під час створення єдиного інформаційного середовища використовував реляційні бази даних для вирішення проблем управління фінансами, організаційної структури, персоналу, навчального процесу, забезпечення навчальними матеріалами тощо. База даних, орієнтована на документи, ведення групової роботи та управління документацією університету. В університеті використовуються дво-

та трирівневі програми клієнт-сервер, компоненти додатків, що використовують технологію CORBA[14], веб-додатки, які використовують Oracle Server[15]. Географічні інформаційні системи MapObject[16] використовуються для завдань, пов'язаних з метрикою. VDUES має два відділення, в яких потрібно впроваджувати ті самі системи управління, що і в головному університеті, і дані повинні бути інтегровані.

Висновки до розділу 1

В сучасному світі відбувається швидкий розвиток інформаційних технологій та їх швидке проникнення у всі сфери суспільного життя, зокрема у галузі освіти, створюючи передумови для переходу країни до більшого рівеня розвитку. Один із напрямків сучасної державної політики України в освіті полягає в удосконаленні інфраструктури інформаційного освітнього простору це відображено в Законі України “Про Національну програму інформатизації”. Останнім часом значна увага приділяється модернізації соціальної та економічної сфер. Це не лише питання розробки певних механізмів взаємодії та реалізації низки державних програм, а ефекту, пов'язаного з їх реалізацією. Орієнтація на економіку знань в цьому плані викликає багато різних питань у галузі освіти, оскільки ця сфера є основною “кузнею кадрів” нового суспільства. Запровадження автоматизованої системи управління вищим навчальним закладом у сферу освіти є елементом державної політики. Головним гальмуючим чиником розвитку інформатизації вищого навчального закладу є недостатнє матеріальне забезпеченна ВНЗ. Але підвищивши кількість персональних комп’ютерів та підвищивши швидкість мережі можливо досягнути більшої ефективності всього ВНЗ. Приклад інформатизіції вищого навчального закладу був розобраний в розділі про типові інфраструктури для систем управління ВНЗ. Було наведено приклад успішної автоматизації процесів в вищому навчальному закладі.

РОЗДІЛ 2.
ПРОЕКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ВИЩИМ НАВЧАЛЬНИМ ЗАКЛАДОМ

2.1. Особливості створення системи управління ВНЗ.

Діяльність сучасного університету є багатопрофільною, а управління університетом на основі інформаційних технологій є складним завданням, яке потребує вирішення організаційно-технологічних проблем з урахуванням економічної доцільності. Можна з певним ступенем умовності визначити кілька проблемних напрямків або контурів інформатизації університету - адміністративний менеджмент та управлінський облік, фінанси, управління навчальним процесом, управління інформаційними ресурсами, власне навчальний процес, дослідження. Як правило, інтеграція засобів інформатизації у кожному колі базується на створенні корпоративного інформаційного середовища університету з метою забезпечення єдності навчальних та управлінських процесів в університеті, а також застосування універсальних методів доступу до інформації. Якщо ми говоримо про управління університетом, то для нього інформаційні технології (ІТ) - це головний інструмент, який створить переваги в конкурентному середовищі. У зв'язку з цим ключовими заходами розвитку ІТ є створення надійної та ефективної інформаційної інфраструктури, впровадження унікальних методів доступу до корпоративних даних, поліпшення керованості всього комплексу інформаційних ресурсів та забезпечення відповідності ІТ-інфраструктури стратегічним цілям. Комплексне застосування цих заходів може бути пов'язане з формуванням корпоративного інформаційного середовища (СНД) університету, що забезпечить інтеграцію інформаційних ресурсів та створить інформаційну інфраструктуру університету відповідно до існуючої організаційної структури та прийнятих правил ведення бізнесу.

Створення системи управління університетом передбачає комплекс заходів щодо впровадження інформаційних технологій у всіх сферах діяльності університету як сукупності організаційних заходів, програмного та апаратного

забезпечення, а також прийомів, методів і прийомів їх застосування при зборі, зберіганні, обробці, передачі та використанні інформації. Ми можемо виділити наступні основні завдання, застосування яких спрямоване на формування інформаційного середовища:

1. Формування організаційної структури інформатизації.
2. Створення університетської інформаційної інфраструктури.
3. Інформатизація навчального процесу.
4. Інформатизація управління університетом.
5. Інформатизація досліджень та проектів.
6. Підвищення рівня компетентності персоналу в галузі інформаційних технологій.

Початкові інвестиції повинні бути зосереджені на створенні інформаційної інфраструктури (корпоративна комп'ютерна мережа, організація доступу до телекомунікацій до регіональних та глобальних мереж даних, розгортання корпоративних серверів, облаштування аудиторій, робочих місць для викладачів та персоналу, застосування інформаційних служб та середовищ) та підвищення рівня компетентності персоналу. ІТ через навчання та грамотну політику щодо освіти та впровадження.

Приймаючи рішення про створення СНД, необхідно уникати руйнівних процесів у діяльності університету, забезпечуючи при цьому розумну кількість інновацій, як у викладанні, так і в управлінні. Це може зробити:

1. Пріоритетне включення до плану інформатизації критичних і вимагає більшості університетських завдань, реалізація яких неможлива без впровадження інформаційних технологій.
2. Система не має бути створена під керівництвом одного або двох менеджерів.
3. Розподіл відповідальності за впровадження ІТ між керівництвом, програмістами та викладачами (працівниками).

4. Удосконалення розробки методів і технологій вирішення завданьу правління та навчання, спрямованих на використання ІТ.
5. Виділення необхідних ресурсів (матеріальних та людських) для всебічного передпроектного опитування, аналізу та технічного проектування компонентів СНД.
6. Стандартизація та документування всіх етапів інформатизації.
7. Управління реорганізацією структури та бізнес-процесів в університеті, що є наслідком переінженерних завдань управління.
8. Створення в колективі всіх рівнів міцної віри у неминучість переходу до сучасних інформаційних технологій.

Створення та організація життєвого циклу СНД - це складне організаційно-технологічне завдання, яке визначає доцільність поетапного розвитку системи, коли на кожній фазі поставляється готовий продукт (обертання СНД), який буде послідовно модифікований та вдосконалений від фази до фази. Взаємозв'язок підсистем та інтеграція даних реалізується на основі організаційної, функціональної, технічної, програмної та інформаційно-мовної сумісності. Тільки за таких умов можна забезпечити стійке функціонування СНД як успішне організаційно-технологічне рішення.

Характеристика університету як об’єкта інформатизації пов'язана з багатопрофільним характером діяльності, великою кількістю форм і методів навчальної роботи, просторовим розподілом інфраструктури (філії, представництва), різними джерелами фінансування, розвиненою структурою допоміжних підрозділів та служб (будівництво, виробництво, бізнес). адаптація до мінливого ринку освітніх послуг, потреба в аналізі ринку праці, відсутність звичайної формалізації ялинового процесу, потреба в електронній взаємодії з батьком м організацій, часті зміни в статусі співробітників і студентів. Певною мірою це полегшує проблему, що університет - це стабільна, ієрархічна система функцій управління, яка має всі необхідні умови життя та функціонує на принципах централізованого управління (останнє означає, що адміністративні засоби можна

активно використовувати для управління завданнями ІТ). Вищезазначені характеристики вимагають дотримання наступних вимог:

1. Комплексне опрацювання завдань щодо інформатизації, починаючи від концепції та закінчуючи підтримкою програмних та апаратних рішень.
2. Залучення великої кількості експертів, які мають значну частину бізнес-процесів.
3. Використання модульної структури бізнес-додатків, коли кожен модуль охоплює взаємопов'язану групу бізнес-процедур або інформаційних послуг, забезпечуючи при цьому унікальні вимоги до інтерфейсів.
4. Застосовуйте розумну послідовність фаз у вирішенні проблеми інформатизації (не приймайте все відразу, потрібно вибрати пріоритетні області та вирішити проблеми до кінця).
5. Розробка документації на основі раціонального використання стандартів, що гарантує створення успішної системи.
6. Використання надійних та масштабованих апаратних та програмних платформ і технологій для різних цілей (Інтернет-технології, веб-сервіси, розподілені обчислення, кластеризація).

Основні заходи щодо створення ефективного та успішного університету:

1. Планування - визначення стратегії розвитку ІТ в університеті в цілому, оцінка інформаційної зрілості університету.
2. Проектування - аналіз бізнес-процесів, бізнес-процедур в управлінні, форм і моделей навчання, пояснення методів та архітектури, методів адміністрування та захисту даних.
3. Створення - розробка організаційних заходів, програмних та апаратних рішень, інформаційного середовища, методів, тестування та налагодження.
4. Документація - підготовка технічної та робочої документації, реєстрація методів.

5. Впровадження - навчання користувачів, впровадження програмно-апаратних рішень у роботу, наповнення баз даних та інформаційного середовища, інтеграція додатків, сполучення з інформаційними системами регіонального та федерального рівня.
6. Підтримка - усунення недоліків, помилок, коригувань, адміністрування, інформаційне та методичне забезпечення персоналу.

З точки зору архітектури в корпоративному інформаційному середовищі можна виділити три рівні:

1. Комп'ютерне мережеве обладнання, канали та лінії передачі даних, робочі станції користувача, системи зберігання даних.
2. Операційні системи, мережеві послуги та ресурси управління доступом, програмне забезпечення середнього рівня.
3. Прикладне програмне забезпечення, інформаційні послуги та орієнтовані на користувача середовища.

Створюючи СНД, необхідно забезпечити міжрівневу координацію відмінностей між вимогами до обраних рішень чи технологій. Так, на іншому рівні архітектура СНД багатьох університетів є різними і погано пов'язаними підсистемами з різними операційними середовищами, узгоджуються між собою лише на рівні закріплення IP-адрес або надсилання повідомлень. Причини поганої системної організації СНД - відсутність затвердженої архітектури СНД, наявність декількох центрів відповідальності за розвиток технологій, які працюють непослідовно. Проблеми починаються з небажання керувати вибором операційних середовищ у відділах, коли ключові технологічні рішення повністю децентралізовані. Університети, які мають чітку стратегію розвитку IT, загальні вимоги до інформаційної інфраструктури, політику інформаційної безпеки та затверджені положення про основні компоненти СНД, як правило, мають міцне адміністративне ядро в управлінні та високий авторитет керівника IT. Такі університети можуть, звичайно, використовувати різні операційні середовища або

системи середнього рівня, але це з організаційних, технічних чи економічних причин і не перешкоджає розгортанню університетів СНД та запровадженню єдиних принципів доступу до інформаційних ресурсів.

2.2. Вибір технології для стоворення Frontend рішень

React[17] - це інструмент для створення інтерфейсів користувача. Його головне завдання - забезпечити, щоб те, що ми бачимо на веб-сторінках, відображалося на екрані. React значно спрощує створення інтерфейсів, розбиваючи кожну сторінку на невеликі шматочки. Ми називаємо ці фрагменти компонентами.

Сам React не дозволить вам створити веб-додаток, оскільки він призначений для створення представлення (тому це MVC "V"). React може створити представлення компонентів, для яких дані можна передати ыншим додаткам. Щоб заповнити цю прогалину, Facebook розробив Flux[18], який є архітектурним шаблоном, який доповнює React. Архітектура Flux у поєднанні з React дозволяє наступний сценарій:

1. Користувач натискає на елемент React.
2. Дія активована. Він надсилає цю дію до сховища через диспетчерські бібліотеки.
3. Репозиторій відстежує стан додатків та методи пошуку даних. Кожне оновлення статусу відображається у представленнях даних, що допомагає підтримувати перегляди у відповідності зі статусом програми.

Мінуси використання React:

1. На початковій стадії React сповільнюється. Нелегко зрозуміти, як працюють компоненти, а документація - це лабіринт інформації. Теоретично це можна швидко вирішити, якщо над ним працює вся команда.
2. React не підтримує IE8[19] та нижчі браузери і ніколи не буде.

3. Якщо ваш додаток / сайт не насичений великою кількістю динамічних сторінок, вам потрібно буде написати багато коду та вирішити невеликі проблеми.
4. Винайдете колесо. React молодий, тому немає встановленої практики. Чи потрібно вашій програмі випадаючого списку, розміру вікна чи світлого поля? Вам доведеться писати все з нуля.

Процес розробки:

1. Швидкий перехід до компонента або функції: Клацніть клавішею CMD, що знаходиться безпосередньо в JSX[20].
2. Швидкий батьківський перехід: CMD + SHIFT + F (глобальний пошук проекту).
3. Швидкий перегляд батьківського списку: Інструменти реагування.
4. Створіть контрольний список можливих станів кожного компонента (помилка, відсутність даних тощо - повний список).
5. Скористайтеся інструментом налагодження. Метод console.assert також корисний.
6. Робота з маріонетковими даними та ляльковим API (json-server[21]).
7. Використовуйте ті самі дані віртуальної машини для вашої програми, тестів та API підтвердження даних.
8. Переадресуйте об'єкт у функцію та використовуйте подрібнювач для отримання названих параметрів. Це набагато простіше читати. Ви можете знищити функцію безпосередньо в підписі функції, що дозволяє негайно документувати очікувані параметри.
9. Розробка на основі історій Створіть і протестуйте кожен компонент ізольовано. Кожен стан документа - це різна історія. Потім використовуйте для знімків.

Обробка даних У React є два типи даних: реквізит та статус. Різниця між ними на початку трохи підступна для розуміння, тому не хвилюйтесь, якщо це вас

бентежить, все стане набагато простіше, коли ви почнете працювати з ними. Ключова відмінність полягає в тому, що умова є приватною і може відрізнятися від самого компонента. Опори зовнішні і не контролюються самим компонентом. Він передається з компонентів вгору по ієрархії, які також контролюють дані. Компонент може безпосередньо змінювати внутрішній стан. Але він не може змінити реквізит так само. Давайте докладніше розглянемо реквізит. Отже наш компонент Hello є статичним і передає те саме повідомлення. Важливою частиною роботи з React є її повторна використання, що означає можливість написати компонент один раз, а потім використовувати його в різних випадках, таких як показ різних повідомлень. Щоб досягти такого виду повторного використання, тепер додамо реквізит.

State - інший спосіб зберігання даних у React - це компонент стану, тобто внутрішній стан. І на відміну від реквізиту, який компонент не може змінити безпосередньо, він має умову Якщо ви хочете змінити дані у вашій програмі - наприклад, на основі взаємодії користувачів - вони повинні знаходитися у стані компонента всередині програми. Щоб ініціалізувати стан, просто встановіть this.state у методі класу конструктор. Наша країна насправді є об’єктом, який у нашому випадку має лише один ключ, який називається повідомленням.

AngularJS[22] - це структурована основа для динамічних веб-додатків. Це дозволяє використовувати HTML[23] як мову шаблону, а також розширити синтаксис HTML, щоб ваш додаток був стислим і стислим. Ви також можете зменшити кількість коду, використовуючи ін'єкцію обов'язкових даних та залежностей (що надходять з поля). І все в браузері на базі браузера, який працює з будь-якою серверною технологією.

AngularJS - це те, яким би був HTML, якби він був розроблений для додатків. HTML - це чудова декларативна мова для статичних документів. Він не містить багато для створення програм, тому, будуючи веб-додатки в браузері, він розуміє, що він змушує мене робити те, що я хочу. Розбіжності між динамічними

програмами та статичними документами часто вирішуються наступними способами:

1. Бібліотеки - це сукупність функцій, які часто використовуються при створенні веб-додатків. За потреби розгорніть свій код та функції виклику з бібліотеки. Наприклад, jQuery[24].
2. Фреймоврки - це спеціальна реалізація веб-додатків, де ваш код детально описує логіку. Рамка бере контроль і називає ваш код у тих випадках, коли йому потрібно зробити щось, що стосується додатків. Приклади: нокаут, прорізкор і т.д.

Angular пропонує інший підхід. Створюючи нові конструкції HTML, він намагається зменшити невідповідність між орієнтованими на HTML документами та потребами додатків. Angular навчає браузера нового синтаксису, використовуючи конструкції, які називаються директивами. Можна навести наступні приклади:

1. Посилання даних у форматі {{}}.
2. Управління структурами DOM[25] для повторюваних / прихованих фрагментів DOM.
3. Підтримка форми та перевірка форми.
4. Пов’язання коду з елементами DOM.
5. Класифікуйте HTML на багаторазові компоненти.
6. Повноцінне рішення.

AngularJS позиціонується як повноцінне рішення в розробці веб-додатків. Це означає, що це не просто частина загальної загадки створення веб-додатку, а весь комплекс. Це дає Angular право визначати, як слід будувати програми CRUD. Але в той же час він намагається забезпечити, щоб його думка з цього приводу була лише відправною точкою, яку ви можете легко змінити. Кутове поле включено. Все, що потрібно для створення програми CRUD, - це купа посилань даних, основні директиви шаблонів, перевірка форми, маршрутизація, глибокі посилання, компоненти для багаторазового використання та введення залежності.Тестування:

рівномірні тести, підсумкові тести, борошно, тест-смужки. Початкова програма зі структурою каталогу та тестовим сценарієм як вихідна точка.

Angular спрощує розробку додатків, надаючи розробнику більш високий рівень абстракції. Як і будь-яка інша абстракція, ви повинні платити за неї гнучко. Іншими словами, кут не підходить для кожного використання. Він є кутовим для додатків CRUD(CRUD — (Create Read Update Delete створит, зчитати, змінити, видалити). На щастя, додатки CRUD складають щонайменше 90% усіх веб-додатків. Однак важливо зрозуміти, коли Angular стане хорошим вибором, а коли - ні. Ігри та редактори інтерфейсів - яскраві приклади дуже інтенсивних та складних DOM-маніпуляцій. Ці типи програм відрізняються від програм CRUD, тому вони не підходять для Angular. У цьому випадку може бути щось краще, ніж обладнання, наприклад, jQuery.

AngularJS дзен:

1. Був створений AngularJS з переконанням, що код декларації краще, ніж потрібно, коли потрібно створити інтерфейс користувача та з'єднати компоненти один з одним; З іншого боку, код блокування чудов підходить для вираження ділової логіки.
2. Це гарна ідея розділити маніпуляцію з DOM та логіку застосування. Цей поділ спрощує тестування коду.
3. Дуже дуже добре ставитися до тестування так само, як до написання заявки. Складність тестування багато в чому залежить від структури коду.
4. Чудова ідея - відокремити клієнтську частину програми від сервера. Це дозволяє запустити розробку та повторно використовувати обидві частини паралельно.
5. Рамковій програмі дуже корисно направляти розробників на всю розробку програми: від проектування користувальницького інтерфейсу, написання ділової логіки, до тестування.
6. Завжди добре робити рутинні завдання невідповідними і складними завданнями.

AngularJS процедури звільнення:

1. Реєстрація зворотного дзвінка: Реєстрація функцій зворотного дзвінка зменшує код і не дозволяє лісу бачитись за деревами. Видалення коду шаблону (наприклад, функції зворотного дзвінка) - це завжди хороша ідея. Це значно зменшує кількість коду, який потрібно написати, і дозволяє вам бачити, що робить додаток.
2. Маніпуляція з програмним забезпеченням DOM: Маніпуляція з HTML DOM є основою програм AJAX[26], але це дуже нудно і ненадійно. З декларативним описом того, як повинен змінюватися інтерфейс користувача, залежно від зміни стану програми ви позбавляєтесь від маніпуляцій низького рівня з DOM. Більшість програм, написаних Angular, ніколи не потребуватимуть програмного маніпулювання DOM, хоча ви можете це зробити, якщо бажаєте.
3. Сортування даних від / до інтерфейсу користувача: Більшість операцій у програмах AJAX - це операції CRUD. Потік даних з сервера до внутрішнього об'єкта, а потім до форми HTML, що дозволяє користувачам змінювати форму, перевіряти форму та відображати помилки перевірки, потім повертатися до внутрішньої моделі, а потім на сервер, генерує великий код панелі. Angular позбавляється від більшості цього коду, залишаючи код, який описує весь потік даних у додатку, а не деталі реалізації.
4. Написання тонального коду просто для того, щоб зробити щось подібне: зазвичай потрібно написати багато додаткового коду, щоб отримати просту програму AJAX "Hello World". За допомогою Angular ви можете швидко створити додаток за допомогою служб, які автоматично вбудовуються в додаток у стилі ін'єкції залежності залежної від Гвіса.

Переваги AngularJS:

1. Підтримка Google.
2. Інструменти для розробників.
3. Уніфікована структура проекту.
4. TypeScript[27].
5. Реактивне програмування.
6. Єдиний кадр з коробкою залежної залежності.
7. Шаблони на основі розширень HTML.
8. Shadow DOM крос-браузер з вікна (або його емуляція).
9. Підтримка браузерів для HTTP, службовців.
10. Не потрібно далі налаштовувати. Більше немає обкладинок.
11. Більш сучасний фреймворк, ніж AngularJS (на рівні React, Vue).
12. Велика громада.

Недоліки AngularJS:

1. Вищий поріг входу через спостережувану та залежність від ін'єкцій.
2. Щоб все працювало добре і швидко, вам потрібно витратити час на додаткові оптимізації (не так швидко за замовчуванням, але в багато разів швидше, ніж AngularJS і швидше з кожною новою версією).
3. Насправді, якщо ви плануєте розробити велику корпоративну програму, у цьому випадку у вас немає найсучаснішої архітектури управління державою - вам потрібно додати Redux або інший державний менеджер, щоб ви не промивали мізки.
4. Angular-Univesal має багато підводних каменів.
5. Динамічне створення компонентів не є незначним завданням.

Насправді всі ці недоліки компенсуються власним досвідом розробника. Все, що вам потрібно буде вивчити в Angular, щоб розробити продуктивні та швидкодіючі програми будь-якої складності, описано в наступних поняттях.

Оформлення форми - Щоб розробити справді складні форми, ви повинні знати реактивні форми, а точніше забути про декларативні форми. Ось один хороший приклад (реактивна форма + підтвердження).

Виявлення змін - Оскільки Angular використовує двостороння прив'язка моделі даних за замовчуванням, додатки будуть використовувати її повільніше, тому в деяких випадках потрібно подбати про правильну стратегію виявлення змін. Ви можете переглядати різні проекти OpenSource.

Шаблон - синтаксис шаблону з точки зору абстракції не сильно змінився в порівнянні з AngularJS, тому ми також можемо записувати умови, цикли, прив’язувати модель даних тощо. Все, що вам потрібно, щоб зрозуміти та зрозуміти у кутових пропозиціях, - це те, що є структурними та деклараційними директивами, а також якими є вхідні параметри та вихідні події.

Маршрутизація - це, мабуть, одне з фундаментальних явищ у розробці веб-додатків. Просто важливо зрозуміти, що маршрутизація, як і компоненти, має свій життєвий цикл, якщо ви це розумієте, ви можете писати дійсно класні програми. Варто також згадати: якщо ви повісите модуль на будь-який із шляхів, а не на компонент, відповідальний за показ сторінки на цьому шляху, модуль завантажується на сторінку на вимогу.

Примітки - до речі, багато початківців цього не знають, але це варто зазначити. Декоратори, які широко використовуються для написання програм у Angular, не є різновидом жорсткої магії TypeScript. Декоратори - це специфікація EcmaScript, і коли браузери почнуть їх підтримувати, вони працюватимуть у браузері як оригінальний. Насправді декоратори дуже корисні та забезпечують досить високу читабельність вашого коду. Одним із прикладів є перевірка моделі даних за допомогою декораторів або десеріалізація / серіалізація даних.

Елементи спостереження - насправді слід зазначити, що скоро спостерігаються специфікації EcmaScript і все це буде підтримуватися в браузерах. З точки зору теорії, якщо відкрити концепцію Спостерігача (спостерігача) - це схема поведінкового дизайну. Також відомий як наркомани. Створює механізм класу, який дозволяє отримувати екземпляр об’єкта цього класу сповіщень від інших об’єктів, щоб змінити їх стан і таким чином спостерігати за ними.

Shadow DOM - це інструмент для створення окремого дерева DOM всередині елемента, який не видно зовні, без використання спеціальних методів. Це специфікація W3C. Грубо кажучи, це зручний спосіб створення ізольованих та декількох веб-компонентів. З технічної точки зору, якщо браузери вже підтримують багато концепцій, використовуваних Angular, нам не знадобляться транспілятори та інші будівельні системи; все, що ми писали на Angular, працювало б вдома.

Компонент керує відображенням вигляду на екрані на основі типової Shadow DOM (для створення закритої візуальної поведінки). Компоненти зазвичай використовуються для створення простого гаджета в інтерфейсі користувача, але в той же час вони можуть бути набором ще простіших компонентів всередині них (для збільшення абстракції та створення простих функціональних гаджетів у програмі).

Сприйняття змін, кожен компонент має власний детектор змін, який забезпечує перевірку з'єднань даних, визначених шаблоном.

Ін'єкційна залежність - це структура структурних моделей проектування, в яких один умовно незалежний об'єкт (послуга) відповідає за кожну функцію застосування, яка може потребувати використання інших об'єктів (залежностей), відомих інтерфейсам. Залежності передаються (реалізуються) службі на момент її створення. Директиви дозволяють отримати прямий доступ до DOM ваших елементів.

Інтернет-працівники - підтримка Web Worker призначена для спрощення паралелізації у вашій програмі. Коли ваш додаток запускається, Angular виконує всю основну роботу з обробки вашої логіки в окремих потоках, ядро виконує обчислення у своїй робочій нитці, тоді як інші функції в потоках взагалі неможливо виконати.

HTTP - Найпоширеніший спосіб отримати дані з онлайн-сервісів - через сервіс HttpClient, який доступний для введення залежностей у ваші компоненти. Кутовий HttpClient досить простий. Все, що нам потрібно зробити - це

зателефонувати методу get і надіслати йому URL. Цей метод отримує об'єкт, який можна спостерігати. Цей клас є частиною бібліотеки rxjs, яка використовується в багатьох місцях у Angular.

Vue.js - бібліотека JavaScript для створення веб-інтерфейсів за допомогою архітектурного шаблону MVVM (Model-View-View-Model).

Оскільки Vue працює лише на "рівні презентації" і не використовується для програмного забезпечення середнього та бек-енду, його можна легко інтегрувати в інші проекти та бібліотеки. Vue.js містить широкий функціональний рівень презентаційного шару і може використовуватися для створення потужних веб-сторінок на одній сторінці.

Особливості Vue.js:

1. Реактивні інтерфейси.
2. Декларативна візуалізація.
3. Прив'язка даних.
4. Директиви (усі директиви мають префікс "V-". Значення статусу передається директиві, а атрибути подій html або Vue JS використовуються як аргументи).
5. Логічні шаблони.
6. Компоненти.
7. Організація заходів.
8. Властивості.
9. CSS переходи та анімації.
10. Фільтри.

Основна бібліотека Vue.js 2 дуже мала (всього 17 кБ). Це гарантує, що завантаженість вашого проекту, що здійснюється за допомогою Vue.js, мінімальна, а ваш сайт швидко завантажується. Ви можете завантажити відповідний файл .js за посиланням.

Vue підходить для невеликих проектів, яким потрібно додати трохи реактивності, надіслати форму за допомогою AJAX, відобразити значення при

введенні даних користувачів, облікових даних або інших подібних завдань. Vue легко виміряти і підходить для великих проектів, тому ми називаємо це прогресивною основою.

Vue також чудово підходить для великих застосувань, з одного боку, завдяки своїм базовим компонентам, таким як Router і Vuex. За допомогою Vue ви можете використовувати обидва загальнодоступні API для створення програм та запуску програм на основі сервера. Але Vue найкраще підходить для розробки рішень, які використовують зовнішні API для обробки даних.

За допомогою Vue ви також можете створити блог про спілкування з популярною CMS. Vue.js також чудово підходить для розробки зручних для користувача динамічних інтерфейсів.

Слід зазначити, що React і Vue дуже підходять для обробки так званих безшумних компонентів - невеликих функцій без стану, які приймають елементи вводу та повернення як вихід. Компоненти Vue.js не мають спеціальних вимог до своїх назв, але рекомендується дотримуватися правил W3C для користувацьких компонентів - використовувати малі літери та розділяти дефіси.

Angular - це фреймворк, а не бібліотека, оскільки вона містить чіткі інструкції щодо структури програми, а також має широкий функціонал. Angular - ідеальне рішення для бізнес-додатків, які не потребують аналізу будь-якої іншої бібліотеки або використання додаткових інструментів. React і Vue, з іншого боку, універсальні. Їхні бібліотеки можуть підключатися до всіх типів пакетів, хоча Vue їх мало, оскільки він ще досить молодий.

Ви можете працювати з React або Vue, просто додавши бібліотеку Javascript до вихідного коду. Це неможливо з Angular, оскільки він розрахований на більш складні завдання. Що стосується мікросервісів та мікроприкладних програм, React та Vue забезпечують більший контроль над розміром програм, дозволяючи вибрати лише ті елементи, які потрібні в певних випадках. Вони також пропонують велику гнучкість для переходу від односторонніх програм до мікропослуг, що дозволяє використовувати частини попереднього додатка. Завдяки своїй широкій

функціональності, Angular найкраще підходить для розробки додатків на одній сторінці.

Angular досить обширний. Завдяки своїй широкій функціональності розмір архівованого файлу становить близько 143 к, порівняно з більш простими Vue та React з 23K та 43K відповідно.

React і Vue мають віртуальну DOM (об'єктну модель об'єкта), яка створює копію об'єктного представлення структурного документа і дозволяє працювати з візуальною копією, а не з самим переглядом. Такий підхід допомагає покращити продуктивність кадрів, і тому ваша програма працюватиме швидше. Зокрема, Vue володіє чудовою продуктивністю та глибоким розподілом пам’яті, але всі ці кадри в основному схожі за своїми характеристиками.

На закінчення слід зазначити, що основними перевагами Vue JS є його простота та легкість у вивченні. Замість того, щоб вивчити складну термінологію та інструменти для створення простого додатку, як це відбувається у React, ви можете розпочати розробку відразу. Це робить Vue JS прекрасним вибором для стартапів або будь-якої команди розробників, яка прагне швидко створити високоефективні веб-програми з легким для читання кодом.

2.3. Порівняльна характеристика розглянутих Frontend рішень

Немає відкритих даних про масштаби використання фреймворків, але непрямий аналіз вакансій на веб-сайті real.com дає досить цікаве розповсюдження:

1 місце - ReactJS 78,1 %

2 місце - AngularJS 21 %

3 місце - Vue.js 0,8 %

Дані опитування StackOverflow вже дають дещо інші цифри: 67% голосів перейшли до React, тоді як 42% програмістів проголосували за Angular.

ReactJS виявився найпопулярнішим з усіх. Але чи означає це, що він буде ефективнішим? Зовсім не тому, що за популярністю лежить лише використання середовища в таких проектах, як Facebook, Instagram. Крім того, Vue.js, наприклад,

має найбільше "зірок" на Github. Тому пропоную порівняти ці фреймворки за ключовими характеристиками.

Візуалізація - відображає кінцевий результат. Сучасна архітектура допускає два типи: на стороні клієнта (сторінка надається завдяки потужності комп'ютера користувача) або на стороні сервера.

DOM - Document Object Model - модель об’єкта документа, яка дозволяє читати та змінювати вміст, форматування та навіть структуру html-документів. Кожен фреймворк має свій підхід до обробки DOM, який впливає на візуалізацію кінцевої сторінки, відображеної на екрані користувача.

Vue.js і React створюють копію DOM, обробляють її, а потім порівнюють результат з оригінальною версією. У заключному документі (тобто на екрані користувача) замінюються лише ті частини сторінки, які відрізняються від результатів обробки. Це значно прискорює завантаження та візуалізацію сторінки. Відповідно, зменшується обсяг трафіку, що особливо важливо для користувачів мобільних пристроїв.

Підхід до управління DOM в AngularJS версією 1.x і вище суттєво відрізняється. Ось поділ на два потоки, при цьому браузер (клієнтська частина) несе відповідальність за надання DOM та створення директив, завантаження коду та послуг, загальний потік (серверна частина). Але це не означає, що візуалізація знаходиться на стороні клієнта - візуалізація все ще виконується серверами. Тому SEO оптимізація не спричинить жодних проблем. Пошукові роботи будуть доступні на правильній сторінці індексації.

Архітектурна складова. ReactJS не стосується до чистих фреймфорків. Це свого роду модифікована бібліотека, "заточена" під MVC (Model-View-Controller, де Модель відповідає за надання даних; View - відображає дані моделі користувачеві, а Controller пояснює дії користувача та змушує модель змінювати зміни).

AngularJS та Vue.js вже належать до чистих фреймфорків. Якщо React лежить в основі архітектури проекту, то це визначає:

1. Необхідність пошуку та впровадження додаткових бібліотек для виконання кожного із завдань;
2. Встановлення функціональної частини програми для певної бібліотеки;
3. Труднощі із залученням інших розробників до проекту через різницю бібліотечного набору кожної програми.

Тому для впровадження архітектури знадобиться більше часу. API низького рівня (інтерфейс програмування програм - набір готових команд, докладніше про те, що таке API) вимагає занадто великої конфігурації.

У разі використання готових фреймворків - Vue.js і Angular, більше не виникає проблем з вибором або налаштуванням бібліотек для різних завдань. API високого рівня забезпечує зворотну сумісність для всіх бібліотек. Це дозволить третьому програмісту підключитися до проекту, не вивчаючи архітектури додатків. Популярність повних фреймворків полягає в уніфікації процесів.

React та Vue.js підтримують лише односторонню передачу даних. У той же час об'єкти блокуються в React. Простіше кажучи, кожен об’єкт програми стосується кінцевих процедур, про які користувач не знає до кінця роботи. Однак React підтримує статус копіювання та передачі. Це означає, що ви можете відновити властивості зареєстрованих елементів на іншому пристрої, запустивши програму та повідомивши про стан компонентів. Тому візуалізація буде однаковою, на екранах обох пристроїв буде однакове "зображення". Vue.js працює дещо інакше. JS все ще односторонній, але компоненти працюють із шаблонами, а вихід - чистий html. Існує підтримка JSX, яка спрощує перехід від React та подібних бібліотек. В рамках логіка процесу автоматизована побудовою директив. Для отримання додаткової функціональності достатньо додати необхідну директиву. Наприклад, потрібно зробити блок-елемент посунули в бік. Для цього все, що вам потрібно зробити, - це додати до властивостей об’єкта директиву "перетягування", після чого ви можете перетягнути мишу по екрану.

AngularJS дещо відрізняється логікою процедури. Так, є все, що характерно для Vue.js, але описи взаємодії об'єктів з'являються в службах, що входять до

складу модулів. Модульна архітектура зручніша при розробці великих додатків. Модуль призначений для вирішення декількох завдань подібної функціональності. В результаті розмір кінцевого коду менший і швидкість обробки вище. Підтримка MVVM (Model-View-ViewModel) дозволяє вирішувати різні проблеми в одному розділі програми з одним набором даних. Залежність функцій визначає двосторонню орієнтацію передачі даних. Кожна процедура може ініціювати іншу процедуру.

Зворотна сумісність. Для розробника програми важлива можливість оновлення архітектури для підключення нових модулів і бібліотек.

AngularJS - ідеальний фреймфорк. Повна залежність від попередніх версій та компонентів. Прямий перехід від 4.0 до 5.0 неможливий, вам потрібно буде встановити всі оновлення між версіями. Це призведе до необґрунтованого збільшення обсягу програми. До речі, цікавий факт, кутової версії 3.0 не існує. Версія 2 негайно супроводжується четвертою.

ReactJS Повна сумісність версій. Бібліотеки різних версій можуть бути пов'язані з додатком, а застарілі властивості можна оновлювати за спадщиною.

Vue.js прогресивний фреймворк(за словами головного розробника Vue.js Евана Ю.). Модульна система схожа на React, але включає всі атрибути рамки JS, яка працює з повною зворотною сумісністю.

Звичайно, порівняти фреймворки об'єктивно досить складно. Але аналіз дозволить розробнику вибрати платформу:

1. Якщо вам потрібно швидко вивчити навколишнє середовище, то виберіть між Vue.js та React.
2. Перейдти на Vue.js буде легко як для AngularJS так і для React программістів. Адже тут ми отримуємо чистий html-код, який знайомий всім розробникам. Прийоми, що застосовуються, приблизно такі ж, як і у Angular.

3. Якщо ви плануєте розробити великий проект, тоді варто розглянути AngularJS. Забезпечує максимальну гнучкість і швидкість візуалізації. Величезний досвід інших розробників дозволить вирішити питання, які обов'язково виникають під час роботи над додатком. React буде надто обширним, і поки що не багато флагманських ліній для Vue.js.
4. Якщо в розробці будуть задіяні інші програмісти, Vue.js стане найкращим вибором. Зрештою, цей фреймворк не тільки простий у вивченні, але й дозволяє змінювати додаток, не руйнуючи його архітектури.
5. Якщо проект передбачає багатоступеневе оновлення та розширення функціональності в майбутньому, тоді варто використовувати Vue.js або React для відмінної зворотної сумісності.

2.4. Збірник програмного додатку Webpack

Webpack - це інструмент, який дозволяє, наприклад, компілювати модулі JavaScript в один JS-файл. Webpack також відомий як конструктор модулів.

З великою кількістю файлів він створює один великий файл (або кілька файлів) для запуску програми. Він також здатний виконувати ряд інших операцій:

1. Допомагає зібрати ваші ресурси.
2. Стежить за змінами та повторно виконує завдання.
3. Можливість завантажити JavaScript наступного покоління до старого стандарту JavaScript (ES5) за допомогою Babel, що дозволяє використовувати найновіші функції JavaScript, не турбуючись про те, підтримує їх браузер чи ні.
4. Перенесить TypeScript в JavaScript.
5. Конвертує вхідні зображення в дані: URI.
6. Дозволяє використовувати файл () для файлів CSS.
7. Запускає веб-сервер має вбудований локальний сервер і перегляд в реальному часі.

8. Працює, замінивши гарячий модуль.
9. Розділяє вихідний файл на кілька файлів, щоб уникнути повільної завантаження сторінки через великий розмір файлу JS.

Webpack не обмежується лише інтерфейсом, але також успішно використовується при розробці бек-енд-програм на Node.js.

У Webpack є попередники, від яких він прийняв багато ідей. Основна відмінність полягає в тому, що ці інструменти відомі як виконавці завдань, тоді як Webpack - це не що інше, як конструктор модулів.

Webpack - це більш зосереджений інструмент. Все, що вам потрібно зробити, - це вказати точку входу для вашої програми, і Webpack проаналізує файли та об'єднає їх у один вихідний файл JavaScript, який містить усе, що потрібно для запуску програми.

Програми, написані на JavaScript, постійно складні, тому хорошим рішенням є використання колектора (або пакета). Такі інструменти дозволяють розробникам пакувати, збирати та загалом організовувати всі ресурси, необхідні для проекту. Ви можете використовувати не тільки третю бібліотеку, але і свої файли. Така модульна система дає змогу досягти кращої організації проекту, оскільки виявляється, що він розділений на невеликі модулі.

В даний час Webpack є одним з найпотужніших подібних інструментів. Він є відкритим кодом і дозволяє вирішувати багато різних проблем. Як і інші інструменти для розробників, веб-сайт має свої плюси і мінуси. Почнемо з переваг: це чудово для роботи з додатками на одній стороні. Онлайн-пакет також може виконувати розширений обмін кодом. Завдяки цим та іншим перевагам він зараз є одним із найпопулярніших інструментів розвитку СВ.

Webpack - це дуже гнучкий інструмент налаштування. Щоб почати працювати з ним, вам потрібно ознайомитися з чотирма основними поняттями:

1. Введення означає точку входу, яку веб-пакет використовуватиме для створення внутрішнього графіка залежності. Після введення точки входу веб-сайт зможе зрозуміти, які модулі та бібліотеки пов'язані

безпосередньо та опосередковано. В результаті кожна залежність перетворюється у файли, що називаються пакетами (пакети - їх можна перекласти у вигляді пакунків чи вузлів).

2. Вихід визначає, де веб-сайт повинен розміщувати склад створених пакетів і як він буде називати ці файли (за замовчуванням - ./dist). Ви можете налаштувати цю частину процесу в полі виводу в конфігурації.
3. Завантажувачі дозволяють веб-пакетам обробляти не тільки файли JavaScript, оскільки веб-пакет розуміється тільки JS. Завантажувачі перетворюють усі типи файлів у модулі, які ви можете потім додати до залежного графа вашої програми (і, отже, до пакету).
4. Плагіни - якщо завантажувачі завантажувачів використовуються для перетворення певних типів модулів, плагіни можна використовувати для набагато більш широкого переліку завдань. Щоб використовувати плагін, потрібно використовувати reque () та додати його до набору плагінів. Більшість плагінів можна налаштувати за допомогою налаштувань. Оскільки один плагін можна використовувати кілька разів для різних цілей, вам потрібно створити кілька окремих примірників з новим оператором.

2.5. Мова програмування TypeScript

TypeScript - об'єктивно орієнтована, високорівнева мова програмування. Розроблений Андерсом Халесбергом (розробником C #) у Microsoft. TypeScript - це мова програмування та інструментарій. Це набраний, вдосконалений JavaScript, вбудований у JavaScript. Іншими словами, TypeScript це JavaScript з деякими додатковими функціями.

TypeScript - лише JavaScript. TypeScript починається в JavaScript і закінчується в JavaScript. Typescript бере основні структурні елементи програми від JavaScript. Відповідно, вам потрібно знати JavaScript лише для використання

TypeScript. Для виконання код TypeScript повністю перетворений в еквівалент JavaScript.

TypeScript підтримує інші бібліотеки JS. Композитний TypeScript може використовуватися з будь-яким кодом JavaScript. JavaScript, створений за допомогою TypeScript, може повторно використовувати всі існуючі рамки, інструменти та бібліотеки JavaScript. JavaScript є TypeScript. Це означає, що будь-який дійсний .js-файл можна перейменувати в .ts та об'єднати з іншими файлами TypeScript. TypeScript портативний. TypeScript - це легко завантажувана мова для браузерів, пристроїв та ОС. Працює на всіх платформах під управлінням JavaScript. Її відмінність від аналогової полягає в тому, що TypeScript не вимагає спеціальної віртуальної машини або часу виконання.

Специфікація ECMAScript[28] - це стандартизована специфікація для мови сценаріїв. Опубліковано шість версій ECMA-262. Шоста стандартна версія - кодове найменування "Гармонія". TypeScript підтримує специфікацію ECMAScript6.

TypeScript використовує основні властивості мови специфікації ECMAScript5, офіційну специфікацію JavaScript. Функції TypeScript, такі як Модулі та Орієнтація на класи, відповідають специфікації EcmaScript 6. Крім того, TypeScript включає такі властивості, як загальний підпис та тип підпису, які не є частиною специфікації EcmaScript6.

TypeScript багато в чому кращий за всі аналогові, такі як мови програмування, такі як CoffeeScript і Dart, оскільки TypeScript - це розширений JavaScript. Для порівняння, Dart і CoffeeScript - це нові мови і потребують тривалості мови. Переваги TypeScript:

Компіляція - JavaScript - інтерпретована мова програмування. Тому його слід перевірити на правильність. У цьому випадку, якщо в коді трапляється помилка, це буде виявлено лише в кінці, коли код уже повністю записаний. Вам знадобиться витратити багато часу, щоб знайти клопа. Компілятор від джерела до джерела забезпечує функцію перевірки помилок. Коли TypeScript знаходить синтаксичні

помилки, він створює код і генерує помилки складання. Таким чином, помилки можна знайти перед виконанням коду.

TypeScript має функцію статичного набору тексту та систему набору ключів через TLS (TypeScript Language Service). Тип змінної, що повідомляється без вказівки типу, може бути зроблений через TLS на основі значення. TypeScript підтримує визначення типів існуючих бібліотек JavaScript. Файл визначення TypeScript (з розширенням .d.ts) містить визначення зовнішніх бібліотек JavaScript. Як результат, код TypeScript може містити ці бібліотеки.TypeScript підтримує такі поняття об'єктно-орієнтованого програмування, як класи, інтерфейси, механізм успадкування тощо.

TypeScript базується на трьох компонентах:

1. Мова - включає синтаксис, ключові слова та підписи типів.
2. Компілятор TypeScript - компілятор TypeScript (tsc) перетворює інструкції, написані в TypeScript, у його еквіваленти JavaScript.
3. TypeScript Language Service - Мовна служба забезпечує додатковий рівень навколо основного конвеєра компілятора, який є у редакторів. Мовна служба підтримує стандартний набір стандартних операцій редагування, таких як автоматичне завершення інструкцій, автоматична заміна підписів, форматування тощо.

Після компіляції сценарію TypeScript можна створити файл визначення (з розширенням .d.ts), який виконує функції інтерфейсу для компонентів у складеному JavaScript. Принцип роботи файлів визначення схожий з принципом роботи файлів заголовків у C / C ++. Файли визначення (файли з розширенням .d.ts) дозволяють автоматично завершувати типи підтримки, виклики функцій та змінні для бібліотек JavaScript, таких як jQuery, MooTools тощо.

2.6. Клієнт-серверна архітектура

Архітектура клієнт-сервер використовується у великій кількості мережевих технологій, що використовуються для доступу до різних мережевих послуг. Розглянемо коротко деякі види таких сервісів (і серверів).

Спочатку вони представляли доступ до документів з гіпертекстом за допомогою протоколу HTTP (протокол передачі тексту Huper). Зараз вони підтримують розширені функції, особливо працюючи з бінарними файлами (зображеннями, мультимедіа тощо).

Сервери прикладних програм призначений для централізованого вирішення завдань додатків у певній предметній області. З цією метою користувачі мають право запускати програми виконання. Використання серверів додатків зменшує вимоги до налаштування клієнта та спрощує загальне управління мережею.

Сервери баз даних використовуються для обробки користувацьких SQL-запитів. У цьому випадку СУБД знаходиться на сервері, до якого підключені клієнтські програми.

Файловий сервер зберігає інформацію у вигляді файлів і дозволяє користувачам отримувати доступ до неї. Як правило, файловий сервер забезпечує певний рівень захисту від несанкціонованого доступу.

Проксі-сервер - по-перше, він виступає посередником, допомагаючи користувачам отримувати інформацію з Інтернету, забезпечуючи при цьому безпеку мережі. По-друге, він часто кешує необхідну інформацію на локальному диску і швидко передає її користувачам без необхідності повторного доступу до Інтернету.

Брандмауери, які аналізують та фільтрують мережевий трафік для забезпечення безпеки мережі.

Сервери віддаленого доступу (RAS) ці системи дозволяють з'єднання через дзвінок. Віддалений співробітник може використовувати ресурси LAN, підключаючись до нього за допомогою звичайного модему. Це лише деякі з усіх

клієнтських та серверних технологій, що використовуються в локальних та глобальних мережах.

Клієнти, що характеризуються поняттям товщини, використовуються для доступу до деяких мережевих послуг. Вказує конфігурацію обладнання та програмне забезпечення, доступне для замовника. Дотримуйтесь можливих граничних значень:

Тонка партія - цей термін визначає клієнта, комп’ютерних ресурсів якого достатньо лише для запуску необхідної мережевої програми через веб-інтерфейс. Інтерфейс користувача такої програми розроблений за допомогою статичного HTML (JavaScript недоступний), вся логіка програми виконується на сервері. Для роботи тонкого клієнта достатньо дозволити йому запустити веб-браузер, у вікні якого виконуються всі дії. Через це веб-браузер часто називають універсальним клієнтом.

Товстий клієнт - це робоча станція або персональний комп'ютер із власною дисковою операційною системою і має необхідний набір програмного забезпечення. Клієнти "жирних" звертаються до мережевих серверів насамперед для додаткових послуг (наприклад, доступу до веб-сервера чи корпоративної бази даних).

Також товстий клієнт означає додаток для клієнтської мережі, що працює в локальній операційній системі. Такий додаток поєднує в собі компонент подання даних (графічний інтерфейс користувача) та компонент програми (клієнтська обчислювальна потужність).

Останнім часом все частіше використовується інший термін: багатий клієнт. Багатий - це своєрідний компроміс між товстими та тонкими клієнтами. Як і тонкий клієнт, багатий клієнт пропонує графічний інтерфейс, вже описаний інструментами XML, який включає деяку функціональну функцію клієнта (наприклад, інтерфейс перетягування, вкладки, кілька вікон, меню, що падає тощо).

Висновки до розділу 2

Було розглянуто технології та підходи за допомогою яких був створений додаток і кожен з фреймворків хороший по-своєму, має свої переваги та недоліки. Але додаток був створений за допомогою бібліотеки React тому що: React значно спрощує створення інтерфейсів, розбиваючи кожну сторінку на невеликі шматочки. За допомогою технології React можливо створити веб-додаток, оскільки він призначений для створення представлення (тому це MVC "V"). React може створити представлення компонентів, для яких дані можна передати іншим додаткам. Для обміну даних React використовує Flux, який є архітектурним шаблоном, який доповнює React. Архітектура Flux у поєднанні з React дозволяє швидка та безпечно передавати данні. Як мову програмування було обрано TypeScript тому що він використовує основні властивості мови специфікації ECMAScript5, офіційну специфікацію JavaScript. Функції TypeScript, такі як модулі та орієнтація на класи, TypeScript включає такі властивості, як загальний підпис та тип підпису, які не є частиною специфікації EcmaScript6. TypeScript багато в чому кращий за всі аналоги, такі як мови програмування, такі як CoffeeScript і Dart, оскільки TypeScript - це розширений JavaScript. Для порівняння, Dart і CoffeeScript - це нові мови і потребують тривалі модифікації. Як збірник веб додатку був обраний Webpack який дозволяє компілювати модулі JavaScript в один JS-файл. Webpack також відомий як конструктор модулів. З великої кількості файлів він створює один великий файл для запуску програми. Він також допомагає зібрати ресурси, стежить за змінами та повторно виконує завдання. Можливість завантажити JavaScript наступного покоління до старого стандарту JavaScript (ES5) за допомогою Babel, що дозволяє використовувати найновіші функції JavaScript, не турбуючись про те, підтримує їх браузер чи ні. Переносить TypeScript в JavaScript.

РОЗДІЛ 3.
ОПИС АРХІТЕКТУРИ РОЗОРОБЛЮВАНОЇ СИСТЕМИ.

3.1 Безпека системних даних

Безпека баз даних:

1. Облікові дані та конфіденційну інформацію користувачів зберігається у зашифрованому вигляді.
2. База даних підтримує шифрування збережених даних, для захисту даних на диску.
3. Наданий найнижчий рівень привілеїв для доступу до облікових записів користувачів у базі даних.
4. Зберігання та розповсюдження конфіденційних даних із спеціальним спеціалізованим сховищем ключів.

Розробка:

1. Переконайтесь, що всі компоненти програми перевірені на вразливості для кожної версії, випущеної для виробництва. До них належать O/S, бібліотеки та пакети. Перевірка автоматизована в процесі CI-CD (CI - безперервна інтеграція), CD - безперервна подача - безперервна подача.
2. Приділена однакова увага безпеці середовища розробки, а також безпеці виробничого сервера.

Ідентифікація

1. Всі паролі змішані за допомогою відповідної криптографічної функції, а саме bcrypt.
2. Введені прості, але відповідні правила паролів, які спонукають користувачів вводити довгі унікальні паролі.
3. Використовується багатофакторна автентифікація у всіх службах входу.

Захист від DDoS-атак

1. DDoS-атаки на наші API не завдають шкоди сайту. Як мінімум, захищені вузькі місця API, такі як процедури входу та генерації токенів.

2. Забезпечені розумні обмеження щодо розміру та структури даних користувачів та вимог.
3. Зменшена можливість DDoS-атаки за допомогою глобального кешованого проксі-сервісу, CloudFlare. Він вмикається, коли знаходитесь під атакою DDOS, і зазвичай виконує пошук DNS.

Веб-трафік

1. Використовується TLS для всього сайту, а не лише форми для входу та відповіді.
2. Файли cookie захищені та httpOnly, а діапазон визначений атрибутами шляху та домену.
3. Використовується CSP (Content Security Policy) для запобігання небезпечним зовнішнім зонам.
4. Використовуються заголовки X-Frame-Option, X-XSS-Protection у відповідях клієнта.
5. Використовуйться механізм HSTS для примусового доступу через протокол TLS. Пересилаються всі HTTP-запити до HTTPS на сервері зворотної сумісності.
6. Використовуются маркери CSRF у всіх форматах та нові заголовки відповіді файлів cookie SameSite, яка одноразово фіксує CSRF для всіх браузерів.

3.2. Налаштування середовища для розробки системи

Webpack побудований на Node.js, тому для запуску вам потрібні і Node, npm (Node Package Manager).

Процес запуску:

1. Завантажте та встановіть Node.js з офіційного сайту. Щоб встановити npm, відкрийте командний рядок (наприклад, MacOS Terminal) і запустіть команду: sudo npm install -g.
2. Запустити команду: npm init.

3. Створюючи файл package.json, просто натисніть "Enter": налаштування за замовчуванням прийнятні. Крім того, ви можете змінювати їх за потребою. Під час створення пакета .json ви побачите невеликий документ із налаштуваннями. Він містить усе, що проект повинен знати про себе.
4. Виконайте наступний код, щоб додати Webpack до проекту. npm install webpack -D
5. Ім'я Webpack буде додано в package.json як devDependency (воно буде використовуватися під час розробки, але не у виробничій версії). Для цього встановіть перемикач -D. До каталогу буде доданий новий каталог модулів node_modules. Веб-пакет (файли модулів) з'явиться в каталозі node_modules.
6. Встановити Docker. Docker доступний в офіційному сховищі Ubuntu, не завжди є останньою версією програми. Краще встановити останню версію Docker, завантаживши її з офіційного сховища Docker. Для цього додайте нове джерело дистрибуції, введіть ключ GPG із сховища Docker і переконайтесь, що завантажена версія є дійсною, а потім встановіть дистрибутив.
7. За замовчуванням постачальник докерів вимагає привілеїв root або групового докера, які автоматично створюються під час встановлення Docker. Коли ви намагаєтеся запустити команду docker, користувач, який не є sudo, або користувач, який не є членом докерної групи
8. Завантажте дистрибутив PHP на офіційному веб-сайті. Виберіть потрібну версію та завантажте ZIP-архів для типу збірки Thread-Safe (щоб він був сумісний із сервером Apache) та відповідну глибину бітів. Наприклад, для цього підручника ми завантажили архів із останньою версією PHP (7.2.12).

3.3. Мікросервісна архітектура додатку.

В основі архітектури додатку був обраний паттерн Flux який в свою чергу базуэться на UDF (Unidirectional Data Flow). Як підхід до розробки був обрабраний мікросервісний підхід. Мікросервісна архітектура - це явна протилежність монолітній архітектурі. При такому підході замість однієї великої програми ми створюємо набір невеликих нещільно пов'язаних та легкозамінних модулів, які взаємодіють. Однією з головних переваг архітектури мікросервісу є можливість використовувати найкращий технічний фонд для кожного окремого завдання.

Крім того, ми можемо виділити інші переваги архітектури мікросервісу порівняно з монолітною:

1. Модульність.
2. Скорочення часу тестування.
3. Скорочення часу впровадження та можливість паралельної реалізації.

Мікрофрентенд - це архітектурний підхід, в якому незалежні програми збираються в одну велику програму. Можна комбінувати різні віджети або сторінки, написані різними командами, з різними кадрами в одній програмі.

Основні переваги мікрофронтового підходу полягають у розробці даної програми:

1. Модульна архітектура. Окремі віджети або сторінки - це абсолютно незалежні програми.
2. Швидкість тестування. Ви можете протестувати зміни на одному гаджеті чи сторінці окремо і лише в цьому додатку, не витрачаючи часу на тестування решти функціональності;
3. Паралельні розгортання. Окремі віджети або сторінки можна і потрібно використовувати самостійно.

Крім очевидних переваг такого підходу, він має важливі недоліки:

1. Збільшити загальну складність програми.
2. Дублювання коду. Кожна програма розробляється окремою командою, яка приймає власні технічні рішення. Це призводить до перезавантаження

одних і тих же фреймів, бібліотек та загального дублювання коду, які можна було б використати повторно.

3. Монолітний пакет JS завжди буде меншим, ніж колекція пакунків у мікрофронтальній архітектурі.
4. Будь-які проблеми з кешем та версіями додатків.
5. Глобальні змінні або стилі CSS - це речі, які слід пам’ятати в архітектурі мікрофронти, програми не є повністю ізольованими.

Застосування цього архітектурного підходу приносить більше проблем та додаткову складність розвитку, ніж переваг. Але великі проекти, поряд з розподіленими групами, з іншого боку, виграють від створення мікрофронтових додатків. Ось чому архітектура мікрофронта широко використовується в даному веб-додатку.

Найкращим підходом у побудові мікрофронтальної архітектури є single-spa. Ось основні причини, через які був обраний підхід single-spa:

1. Підтримка сучасних фреймворків та бібліотек.
2. Хороша документація.
3. Можливість знаходити залежності між окремими будівельними блоками.
4. Підтримка незалежних розгортань.
5. Розробка додатків на стороні клієнта.
6. Екосистема обгортки готується до швидкої інтеграції існуючих додатків в архітектуру мікрофронту.

Single-spa - це фрейиворк, який дозволяє поєднувати різні програми, незалежно від використовуваної бібліотеки чи фреймворка, в одне ціле. Під прикриттям одного single-spa є набір існуючих інструментів разом з нашими власними рішеннями:

1. SystemJS - завантажувач модулів, необхідний для асинхронного завантаження окремих додатків.

2. Обгортки - single-spa забезпечує окремі обгортки для кожного кадру, створюючи обгортку над додатком, необхідним для інтеграції та з'єднання окремої програми у загальний єдиний single-spa.
3. API - single-spa пропонує набір інструментів, необхідних для спілкування між окремими програмами.

Зв’язок між окремими елементами мікрофронтової архітектури, побудований за допомогою єдиного SPA(Single Page Application – односторвнковий додаток) :

Root Application - корінь програми. Тут основний сигл-спа підключений як основний каркас і конфігурація SystemJS для належного завантаження зовнішніх програм.

Для кожної програми інтеграції кожна дочірня програма повинна забезпечувати загальнодоступні способи запуску, встановлення, відключення, що використовуються автоматичним фреймворком SPA для ручного запуску програми. Практично кожен сучасний фреймворк має готову обкладинку, яка спрощує це завдання та автоматизує частину процесу. На веб-сайті single-spa можна знайти список усіх фреймворків, для яких є готові обгортки. Маючи наявну обкладинку для однієї пропозиції SPA для React.js, ми створюємо інтерфейс із методами запуску, монтажу, відключення, де ми адекватно описуємо, як запустити наш додаток. Vrapper допомагає інкапсулювати внутрішнє виконання та створити API для належного підключення до єдиної SPA. Як і у React.js, ми пропонуємо інтерфейс для запуску програми вручну.

Окрім використання спеціальних обгортків, додаток побудований як модуль amd. Кожен такий модуль асинхронно з'єднаний з коренем всієї архітектури мікрофронтових програм - Root Application. Root Application - це простий HTML-файл з базовими конфігураціями для завантаження інших програм. Є можливість зареєструвати багато мікропрограм в одній програмі microfront. Якщо просто вкажете в параметрі методу registerApplication під час реєстрації програми, така програма буде завантажена для кожної адреси доступу. Цей підхід був

використаний для створення навігаційної панелі або частин програми, які є нормальними і не повинні перезавантажуватися під час завантаження програми.

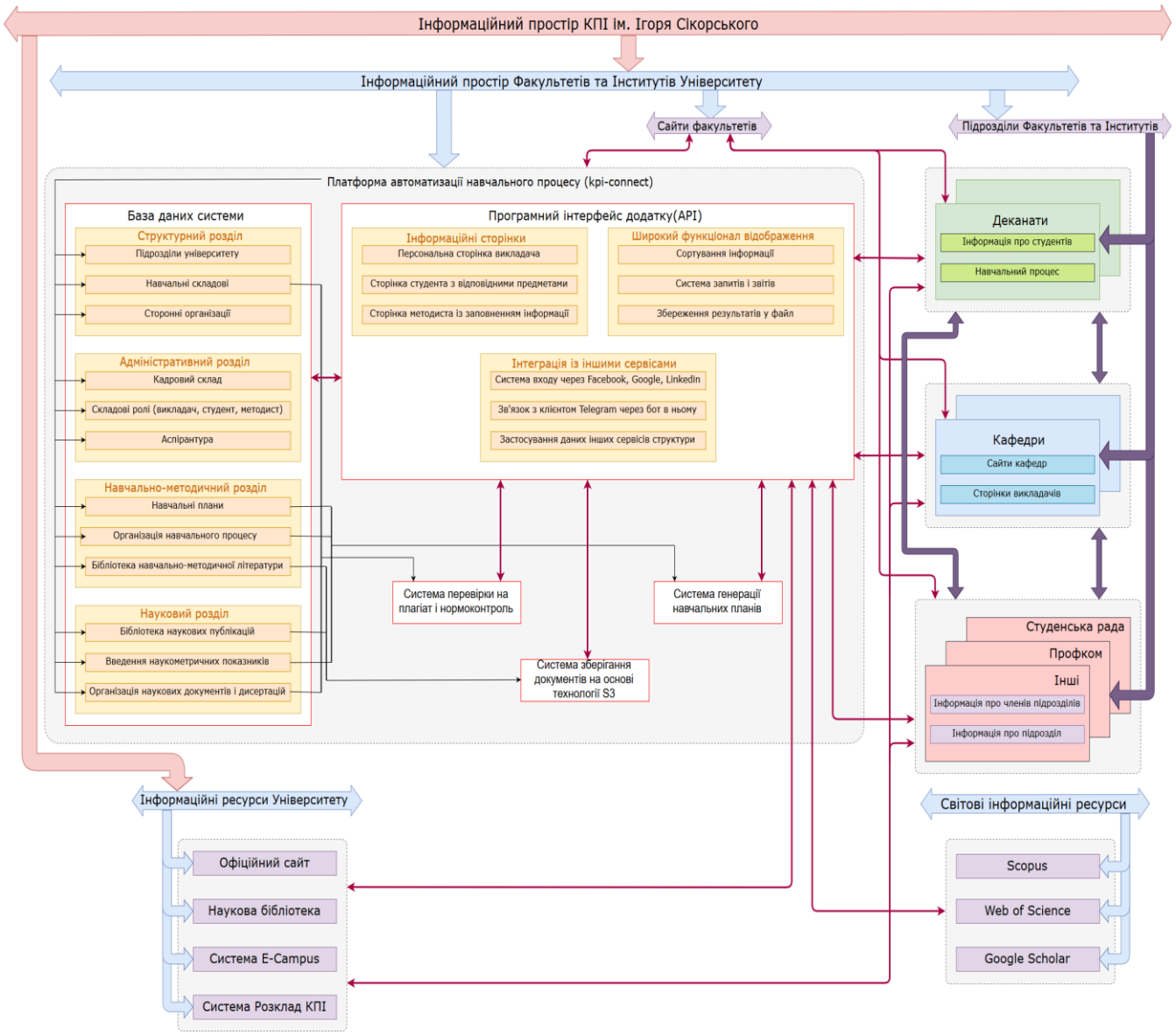


Рис. 3.1. Структурна схема платформи автоматизації навчального процесу

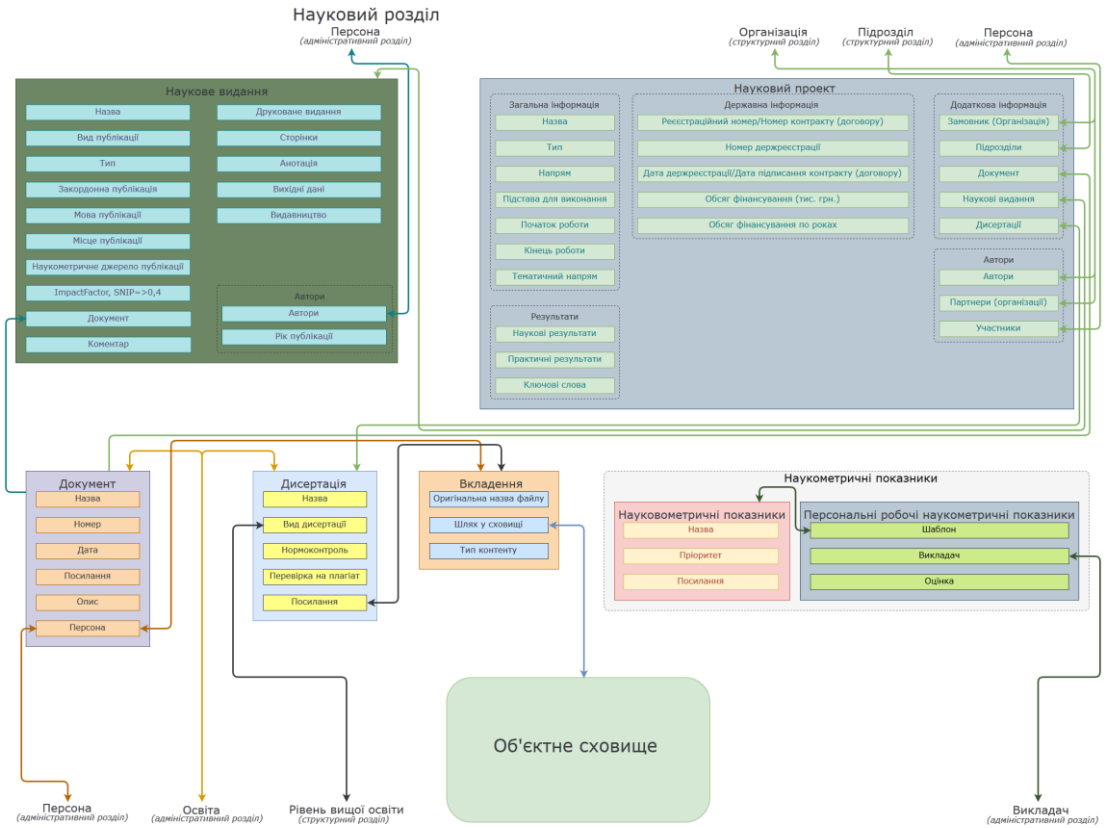


Рис. 3.2. Структурна схема Наукового підрозділу з об’єктним сховищем

3.3 Компонентна архітектура додатку.

Для створення більш ефективних інтерфейсів користувача в React завжди доцільно розбити загальні елементи дизайну, такі як кнопки, форми тощо, на компоненти, що використовуються повторно, з чітко визначеними інтерфейсами. Суть багаторазових компонентів полягає в тому, що ми будемо використовувати їх у багатьох додатках. І ми часто їх використовуємо кілька разів в одному додатку. Тому в нашому додатку краще не використовувати жорсткий код. Тому ми повинні зберігати HTML-ідентифікатори у своїх багаторазових компонентах.

Чому HTML-ідентифікатори повинні бути унікальними? Тому що:

1. HTML-ідентифікатори можуть контрастувати з іншими речами на сторінці, де ваша програма використовує компонент для багаторазового використання.

2. Вставлення жорстко закодованого або статичного HTML в компонент для багаторазового використання може призвести до помилок у нашій програмі. React може надіслати повідомлення про недійсні дублікати ідентифікаторів, якщо програма використовує кілька примірників компонента на одній сторінці.

Натомість компоненти можуть спілкуватися з всією програмою лише через інший пристрій введення, наприклад клавіатуру. Тому важливо, щоб програма відповідала звичайним натисканням клавіш, таким як стрілки.

Користувачі часто використовують клавішу Tab для переміщення між вкладками. Тому програма відрізняє між індексами табуляції та чи встановлюються індекси вкладки в логічній послідовності.

React поставляється з такими елементами, як текст та кнопка. Коли це можливо, використовуйте їх замість div та span. Такі елементи, як текст і кнопка, мають вбудовану правильну поведінку для керування подіями, фокусом та введенням на клавіатурі.

3.4 Управління потоками даних.

Redux став однією з найпопулярніших реалізацій ідей управління потоком даних Flux в додатках React. Однак у процесі вивчення Redux часто виникає ситуація, коли управління та доступ до даних стає надто складним. Щоб уникнути цього був використаний простий і продуманий підхід до програм, що використовує Redux.

У нашої програми є два рівні. Спочатку ми пропонуємо користувачеві вибрати чотири ролі які визначаються в залежності від того під яким логіном заходить користувач. Отже, визначившись з ролью користувача ми можемо надати потрібні йому данні та передати йому відповідний бандл. Нам потрібно буде зберегти дані, отриманих від сервера. Також вам потрібно буде зберегти ідентифікатор користувача, надану користувачу роль. Також підтримується зміна обраної ролі. Наприклад, якщо в нашому випадку вже вибрано роль викладача, а

користувач обирає роль нормокнтролера ми видалимо нинулу роль та надамо новий бандл і дані.

Щоб швидко перезавантажувати дані вони зберігаються в state додатку а Redux має редуктор, конструкцію, яка зберігає стан та оновлює його.

В додатку створений компонент контейнер, який відображає дані доступні для отриманої ролі, як тільки вони з’являються. Цей компонент буде підключений до редуктора, що означає, що компонент розумний, тому що він використовує Redux.

Наш компонент починає свою роботу з DidMount, метод презентації який завантажую початкові дані. Це асинхронний акт і буде виконуватися за допомогою Redux-saga.

Для зручності роботи з API ми створимо нову службу, яка отримує поточний стан мережі. Це асинхронний метод, і ми будемо використовувати його в очікуванні. В додатку використовується API async / wait.

Висновки до розділу 3

В даному розділі було розглянуто які методології та архітектурні підходи були використані при створенні додатку. Для забезпечення безпеки додатку було приняті міри для безпечного зберігання даних а саме облікові дані та конфіденційну інформацію користувачів зберігається у зашифрованому вигляді. Була зменшена можливість DDoS-атаки за допомогою глобального кешованого проксі-сервісу, CloudFlare. Він вмикається, коли сервер під атакою DDOS, і зазвичай виконує пошук DNS. В основі архітектури додатку був обраний паттерн Flux який в свою чергу базуэться на UDF а мікросревісна архітектура додатку надала наступні переваги:

1. Модульність.
2. Скорочення часу тестування.
3. Скорочення часу впровадження нового коду та можливість паралельної реалізації.

Застосування цього архітектурного підходу приносить більше проблем та додаткову складність розвитку, ніж переваг. Але з іншого боку, це надаэ можливысть створення ізольованих мікрофронтових додатків. Ось основні причини, через які був обраний підхід single-spa: Підтримка сучасних фреймворків та бібліотек, хороша документація, можливість знаходити залежності між окремими будівельними блоками, підтримка незалежних розгортань, розробка додатків на стороні клієнта, екосистема обгортки готується до швидкої інтеграції існуючих додатків в архітектуру мікрофронту.

РОЗДІЛ 4
ТЕСТУВАННЯ РОБОТИ СИСТЕМИ

4.1. Опис інтерфейсу системи

Для того щоб ввійти в систему потрібно перейти на головну сторінку додатку.При запуску платформи буде показано головну сторінку, яку зображено на рис. 4.1.

Рис. 4.1 Головна сторінка платформи

Для аторизації у систему необхідно натиснути кнопку ввійти Відповідне вікно входу в систему зображено на рис. 4.2

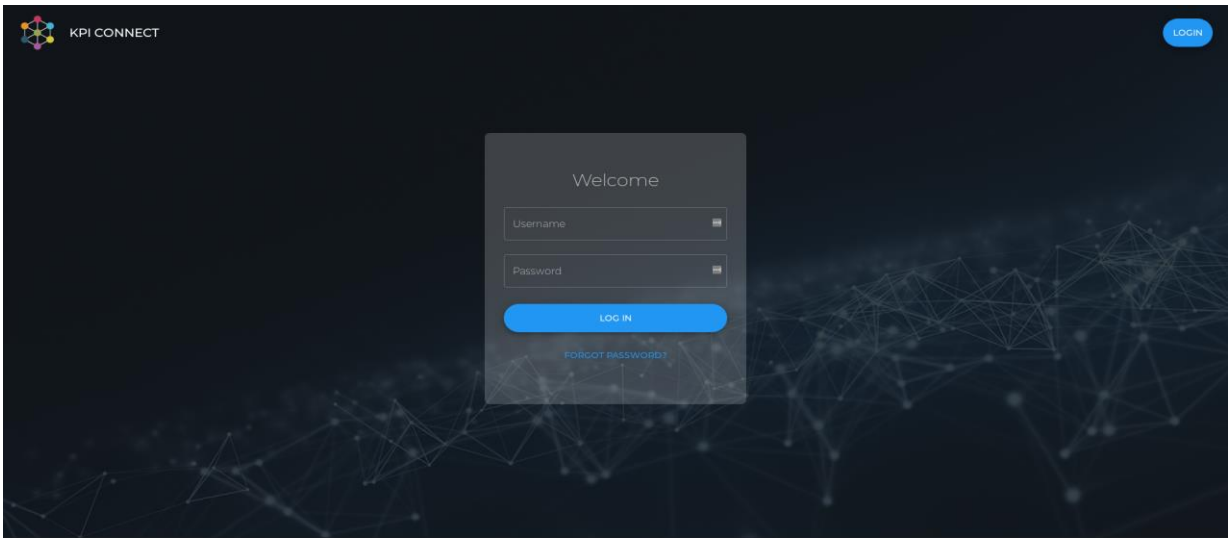


Рис. 4.2. Сторінка входу в систему

4.2. Інтерфейс створення персони

Основою нашої системи є персона, до неї є можливість прив’язати групу, вчений рівень, посаду, наукову роботу.

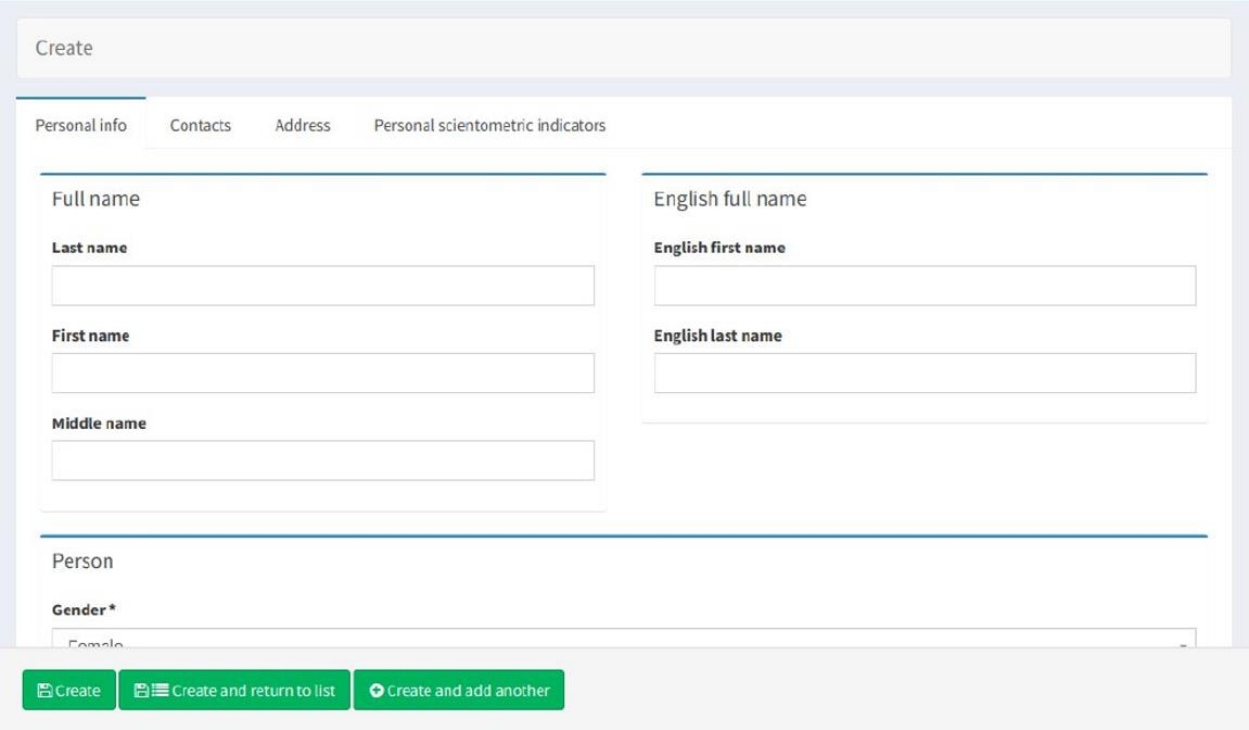


Рис. 4.3. Сторінка створення персони.

4.3. Інтерфейс створення аспіранта

За допомогою даного інтерфейсу є можливість зарахувати персону на аспірантуру привязавши до нього викладача.

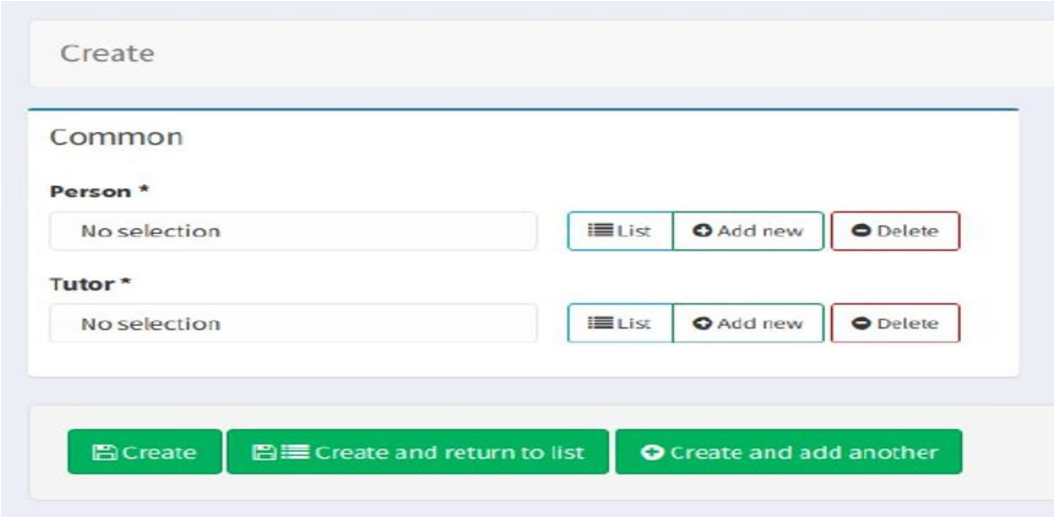


Рис. 4.4. Сторінка створення аспіранта.

4.4. Інтерфейс перегляду даних аспіранта

Переглянути дані створеного аспіранту можна переглянути на сторінці аспіранта.

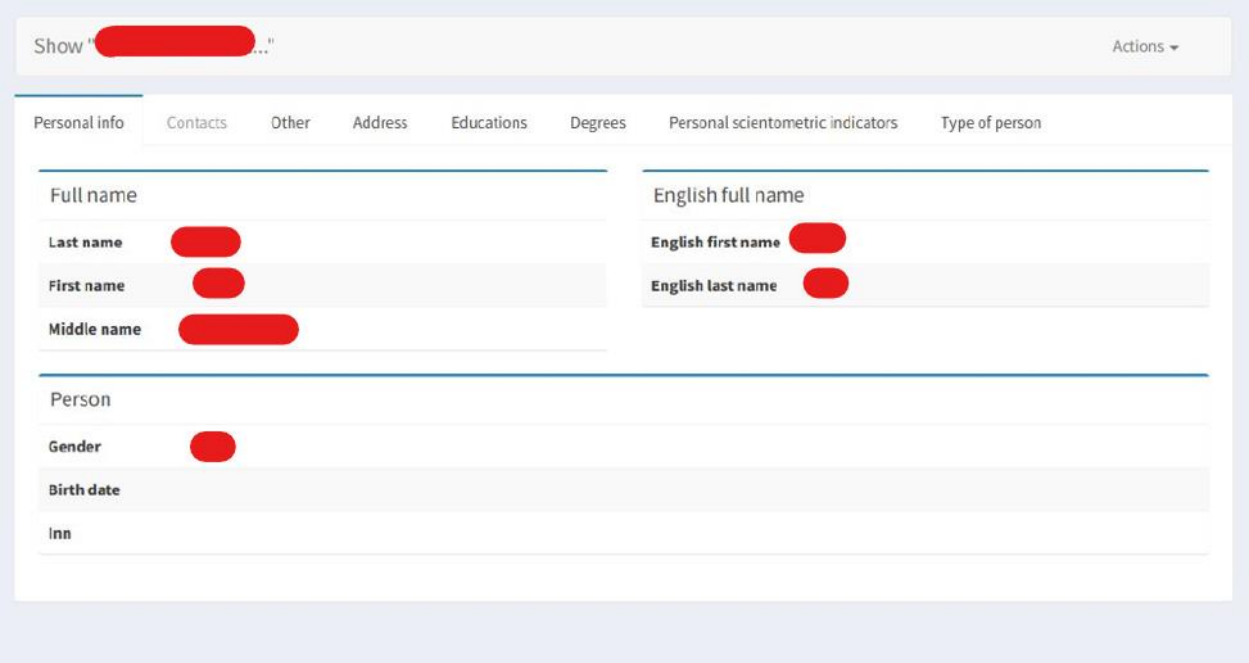


Рис. 4.5. Сторінка даних персони.

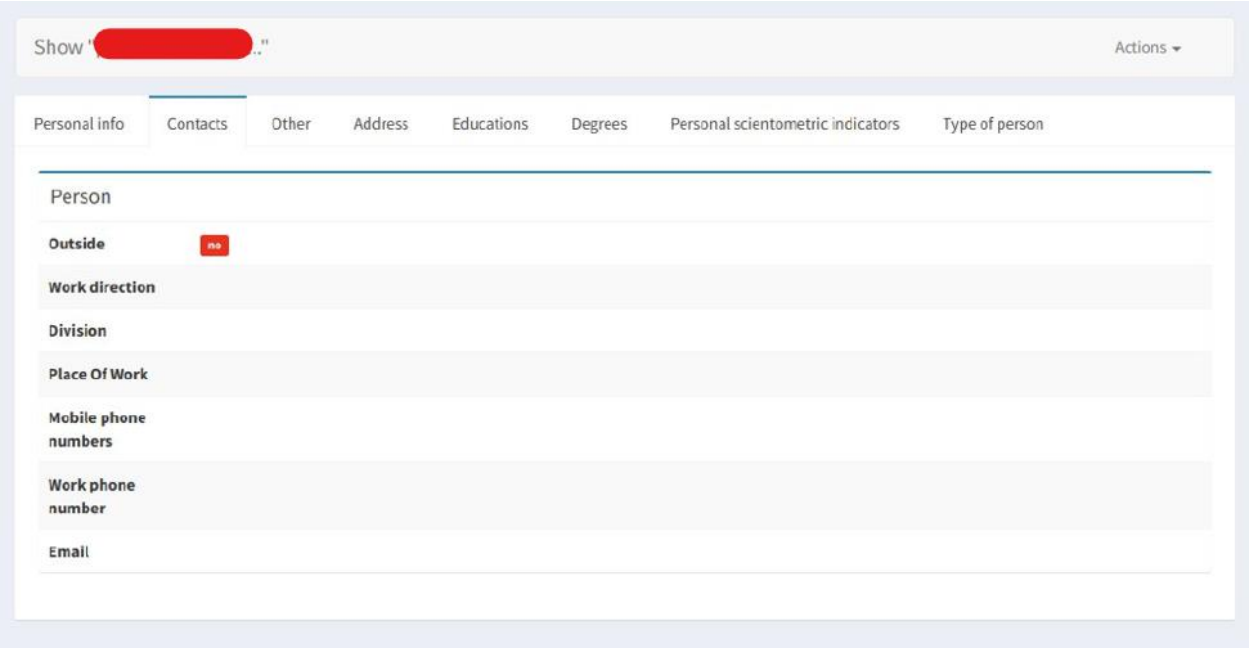


Рис. 4.6. Сторінка контактів персони.

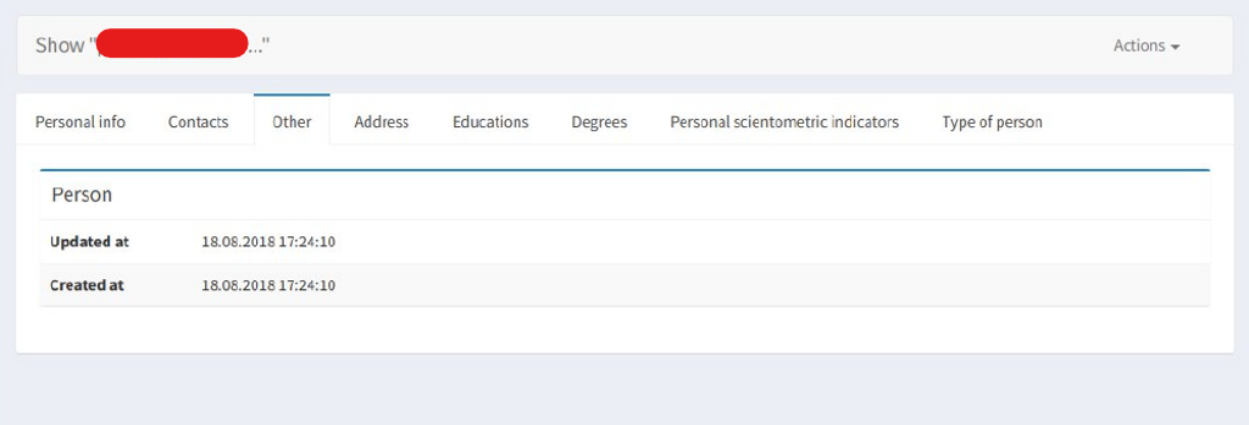


Рис. 4.7. Сторінка додаткової інформації персони.

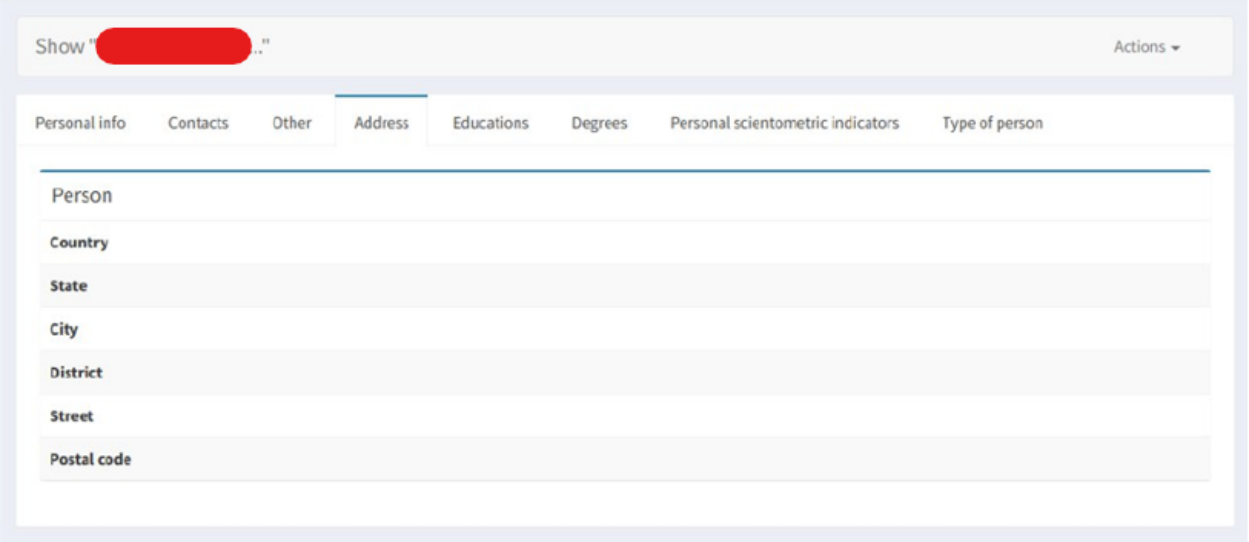


Рис. 4.8. Сторінка інформації про адресу персони.

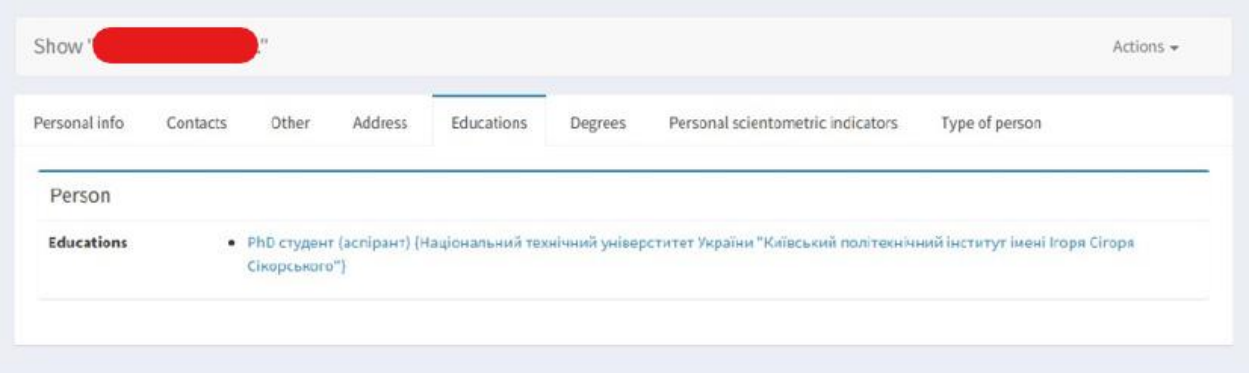


Рис. 4.9. Сторінка інформації про освіту персони.

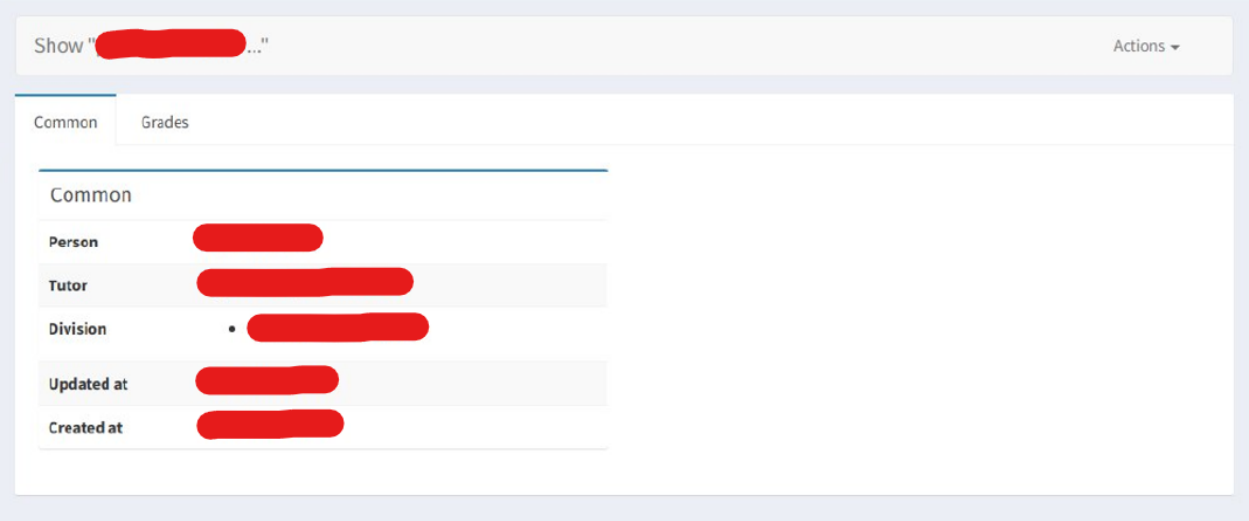


Рис. 4.10. Сторінка даних аспіранта.

4.5. Інтерфейс перегляду даних аспірантів

Переглянути дані всіх аспірантів наявних в системі можливо на цій сторінці, також є можливість відкрити розгорнуту інформацію про аспіранта, відфільтрувати аспірантів по полям, та додати нового аспіранта.

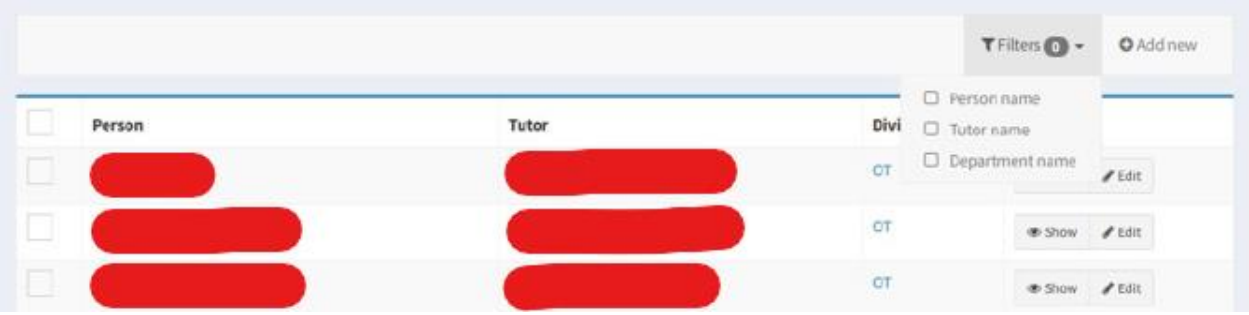


Рис. 4.11. Сторінка даних аспірантів.

4.6. Інтерфейс редагування даних аспіранта

В системі є можливість відредагувати данні існуючого аспіранта

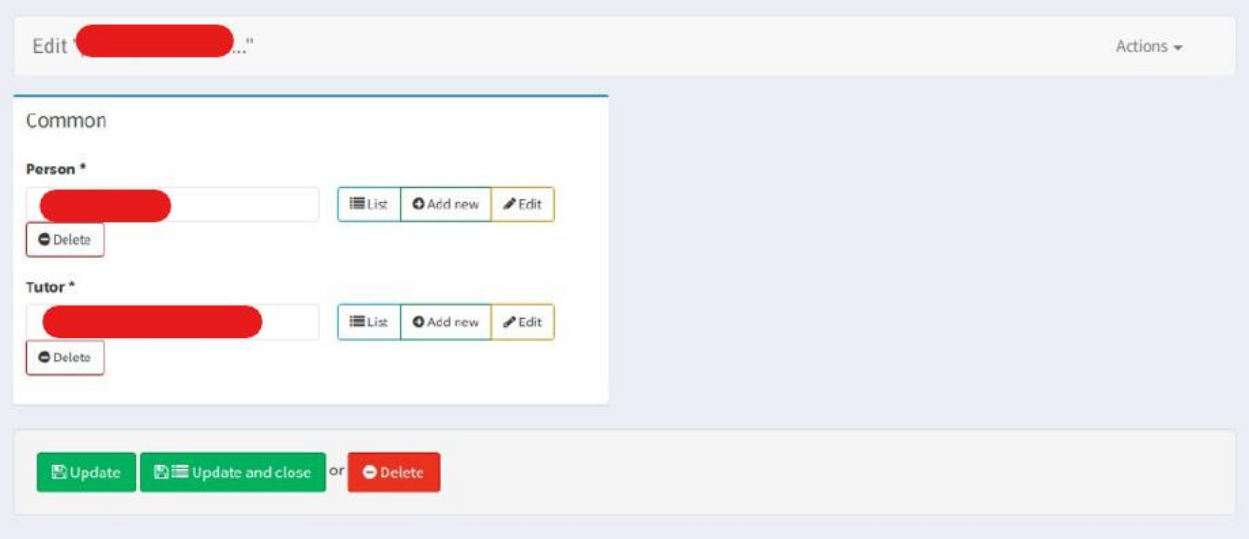


Рис. 4.12. Сторінка редагування даних аспіранта.

Висновки до розділу 4

Тестування роботи системи показує програмний інтерфейс в якій є можливість авторизоватись в системі на головній сторінці веб додатку ввівши свій логін та пароль. Авторизувавшись викладачем або адміністратором система надає графічний інтерфейс в якому є можливість створити нового аспіранта, створити нову персону, зарахувати нову персону на аспірантуру, відредагувати дані про аспіранта чи персону, прив’язати керівника до нового аспіранта. Також є інтерфейс в якому є можливість переглянути всіх аспірантів, відфільтрувати всіх аспірвнтів за запропонованими параметрами, можливість перейти на індивідуальну картку аспіранта, в якій є можливість редагувати данні про аспіранта. В цілому чсистема дуже зручна в викристанні, і значно пришвидшує процесс управління аспірантурою.

Наведене вище тестування показує, що розроблена система працює належним чином, а структура обробки виключень ефективно управляє всіма винятковими ситуаціями, пов'язаними з аспірантурою Можливість легко та просто зараховувати на аспірантуру, редагувати дані про аспіранта значно спрощує процесс управління аспірантурою.

ВИСНОВКИ

В дипломному проекті завершено розробку системи управління аспірантурою в вищому навчальному закладі, що складається з бази даних та веб-додатку для відображення та обробки інформації про аспірантів.

Першим кроком був аналіз проблем розробки платформи автоматизації управління втщтм навчальним закладом. Це включає врахування основних структурних елементів системи, бази даних, інтерфейсу, що дозволить кожному легко та ефективно працювати з даними системи. Для вирішення проблеми взаємодії інтерфейсу пам'яті було обрано API Amazon S3. Це простий та ефективний спосіб скласти логіку завантаження та передачі даних, при цьому усвідомлюючи мінімальну логіку роботи з ним. На основі досліджень була розроблена програмна реалізація платформи, де PHP Symfony був обраний за основну мову програмування як одного з найшвидших та найпотужніших рішень на ринку програмного забезпечення. Користувацький інтерфейс у вигляді фронтального проекту реалізований в JavaScript із бібліотекою React. Інтерфейс додатку відображає сучасний стиль веб-платформи і в той же час робить інтерфейс простим і практичним. Наведене вище тестування показує, що розроблена система працює належним чином, а структура обробки виключень ефективно управляє всіма винятковими ситуаціями, пов'язаними з аспірантурою Можливість легко та просто зараховувати на аспірантуру, редагувати дані про аспіранта значно спрощує процесс управління аспірантурою. Результатом цієї роботи є розроблений програмний продукт для управління аспірантурою в вищому навчальному закладі та автоматизація навчального процесу аспірантів. Дослідження та відповідна архітектура системи підтверджують, що її реалізація можлива, і було використане досить ефективне та потужне рішення. Як показує практика, розроблена архітектура в цілому та окремі компоненти системи є досить перспективним рішенням на ринку програмного забезпечення у сфері освіти країни. Тому подальше розширення функціональності може значно поліпшити кінцевий результат розробленої платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ІНФОРМАТИЗАЦІЯ ВИЩИХ НАВЧАЛЬНИХ ЗАКЛАДІВ ЯК ПРІОРИТЕТ ДЕРЖАВНОЇ ОСВІТНЬОЇ ПОЛІТИКИ В УКРАЇНІ [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: http://www.kbuapa.kharkov.ua/e-book/db/2010-1/doc/2/07.pdf.
2. ЗАКОН УКРАЇНИ Про Національну програму інформатизації [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/74/98-%D0%B2%D1%80.
3. ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА ЇХ ВИКОРИСТАННЯ В УПРАВЛІННІ ВИЩИМ НАВЧАЛЬНИМ ЗАКЛАДОМ [Електронний ресурс]. – 2007. – Режим доступу до ресурсу: https://pidruchniki.com/86566/menedzhment/informatsiyni_tehnologiyi_vikoristannya_upravlinni_vischim_navchalnim_zakladom.
4. Експлуатаційні документи Cisco [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://www.cisco.com/c/ru_ru/about/accompanying-documents.html.
5. Експлуатаційні документи Cisco Neteck [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://www.cisco.com/c/ru_ru/about/accompanying-documents.html.
6. Hardware Guides FireBok 2500 [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: https://www.watchguard.com/wgrd-help/documentation/hardware-guides.
7. Обладнання транспортних sdh мереж [Електронний ресурс]. – 2009. – Режим доступу до ресурсу: https://studfile.net/preview/5206993/page:18/.
8. Синхронные каналы SDH/SONET [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: https://www.opennet.ru/docs/RUS/inet_book/4/43/sdh_436.html.
9. Microsoft documentation [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://docs.microsoft.com/uk-ua/.

10. Freebsd documentation [Електронний ресурс]. – 2001. – Режим доступу до ресурсу: https://www.freebsd.org/ru/docs.html.
11. Open LDAP documentation [Електронний ресурс]. – 1998. – Режим доступу до ресурсу: https://www.openldap.org/doc/.
12. Oracle documentation [Електронний ресурс]. – 2004. – Режим доступу до ресурсу: https://docs.oracle.com/en/.
13. Europe 2020 indicators - Lithuania [Електронний ресурс] // Eurostat. – 2020. – Режим доступу до ресурсу: https://ec.europa.eu/eurostat/statistics-explained/index.php/Europe_2020_indicators_-_Lithuania.
14. Common Object Request Broker Architecture [Електронний ресурс]. – 2002. – Режим доступу до ресурсу: https://www.corba.org/.
15. Oracle Servers Documentation [Електронний ресурс]. – 2009. – Режим доступу до ресурсу: https://docs.oracle.com/en/servers/.
16. Map Object Java Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/javase/8/docs/api/java/util/Map.html.
17. Початок роботи React [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://uk.reactjs.org/docs/getting-started.html.
18. Flux documentation [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://docs.fluxcd.io/en/1.19.0/.
19. Explorer 8 browser changes [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/win32/win7appqual/appendix-1--internet-explorer-6-to-internet-explorer-8-browser-changes.
20. Вступ до JSX [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://uk.reactjs.org/docs/introducing-jsx.html.
21. JSON Server [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.npmjs.com/package/json-server.
22. AngularJS Documentation [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://angularjs.org/.

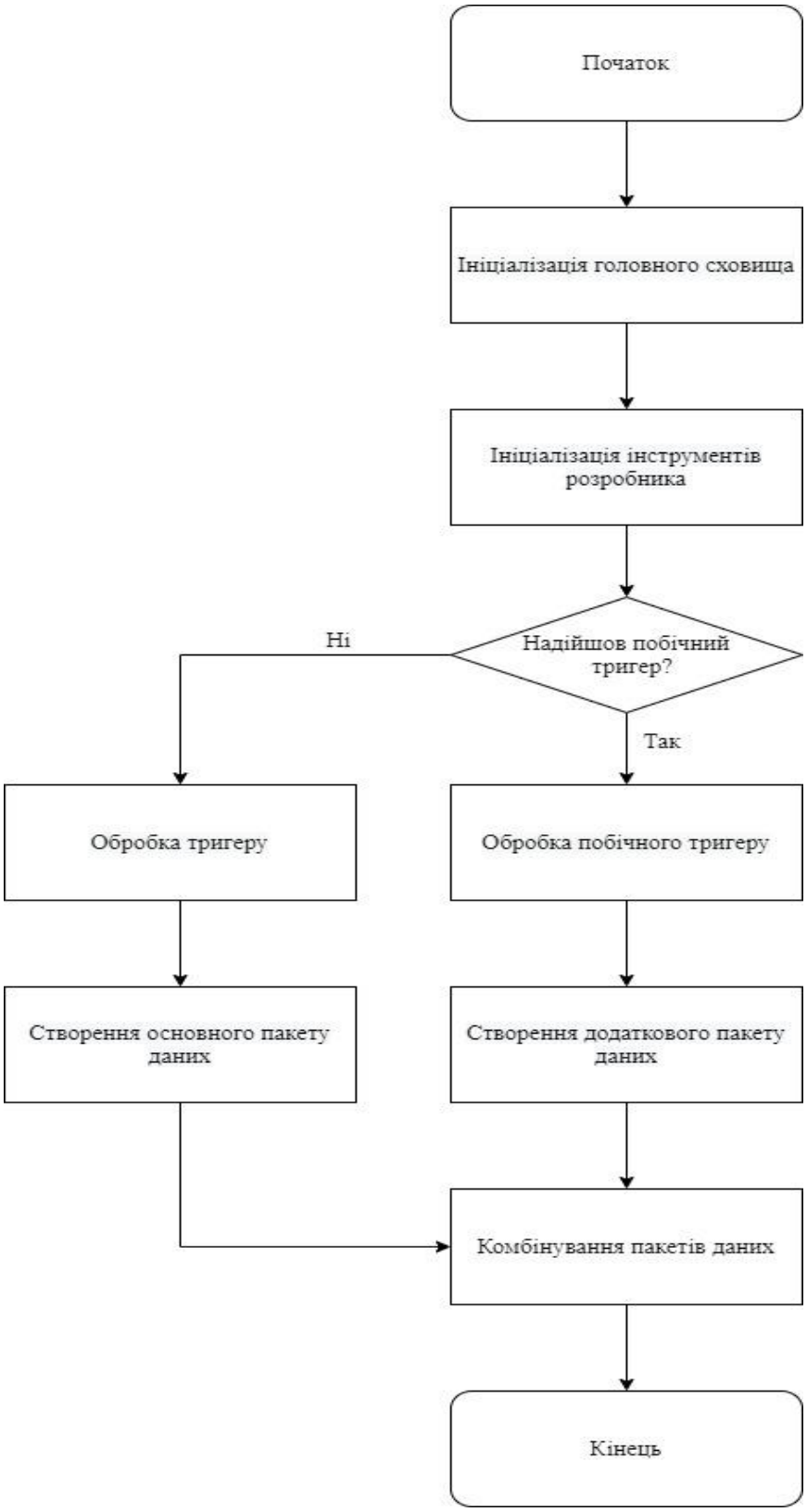
23. HTML Documentation [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: https://devdocs.io/html/.
24. jQuery API [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: https://api.jquery.com/.
25. Руководство по DOM [Електронний ресурс]. – 2007. – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/DOM/DOM_Reference.
26. AJAX Jquery API [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://api.jquery.com/category/ajax/.
27. Typescript Language Documentation [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.typescriptlang.org/docs/home.html.
28. ECMAScript Language Specification [Електронний ресурс]. – 2002. – Режим доступу до ресурсу: https://www.ecma-international.org/ecma-262/5.1/.

ПРИНЦИПОВА СХЕМА

до дипломного проєкту

На тему “Система забезпечення якості освітньо-наукової діяльності аспірантури”

ПРИНЦОПОВА СХЕМА



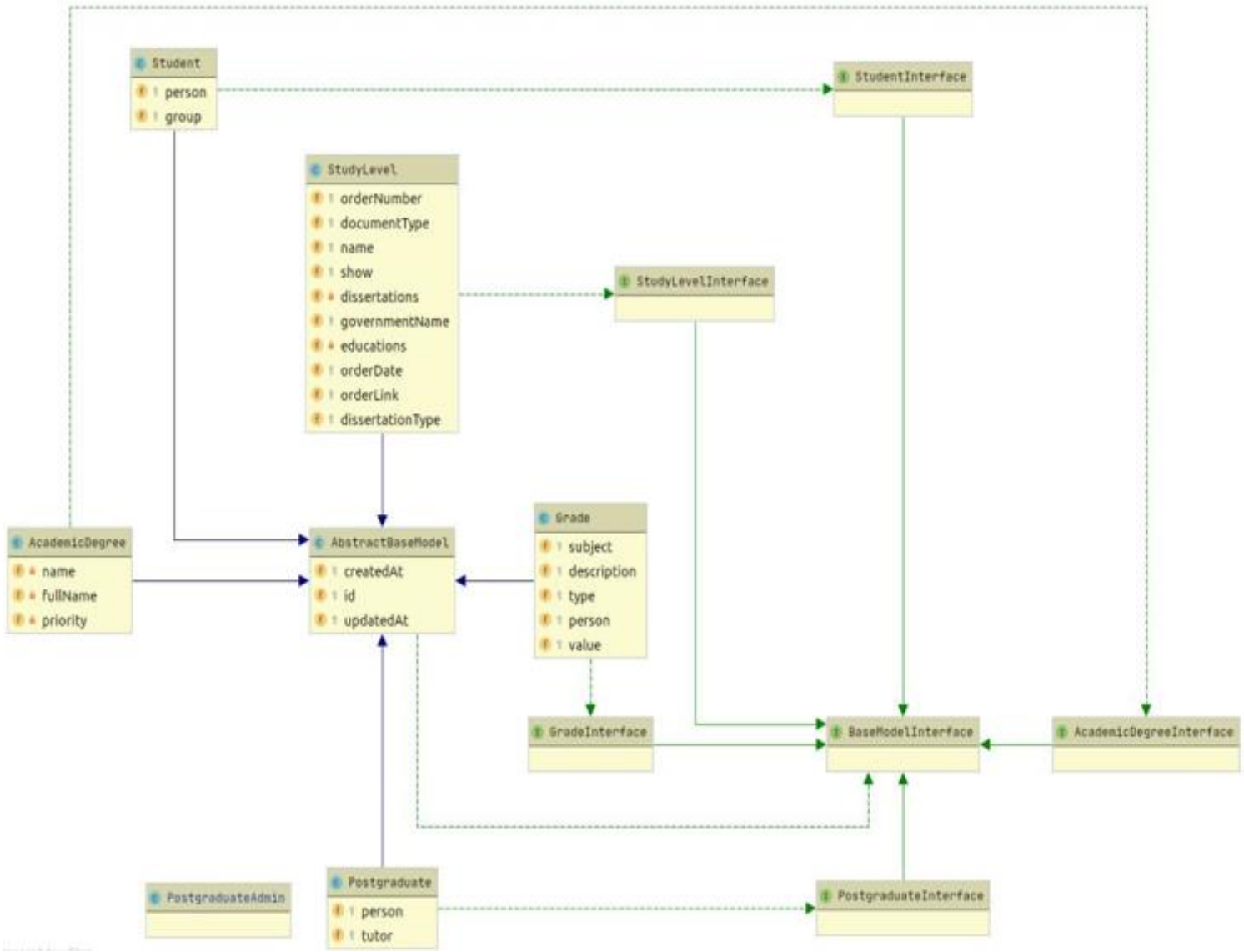
					ДП.6408.04.000 Д1		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив					*Система забезпечення якості освітньо-наукової діяльності аспірантури* Пояснювальна записка	Літ.	Аркуш 1 / Аркушів 1
Перевірив							
Реценз.						НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІО-64	
Н. Контр.		Сімоненко В. П.					
Затвердив							

ФУНКЦІОНАЛЬНА СХЕМА

до дипломного проєкту

На тему “Система забезпечення якості освітньо-наукової діяльності аспірантури”

ФУНКЦІОНАЛЬНА СХЕМА



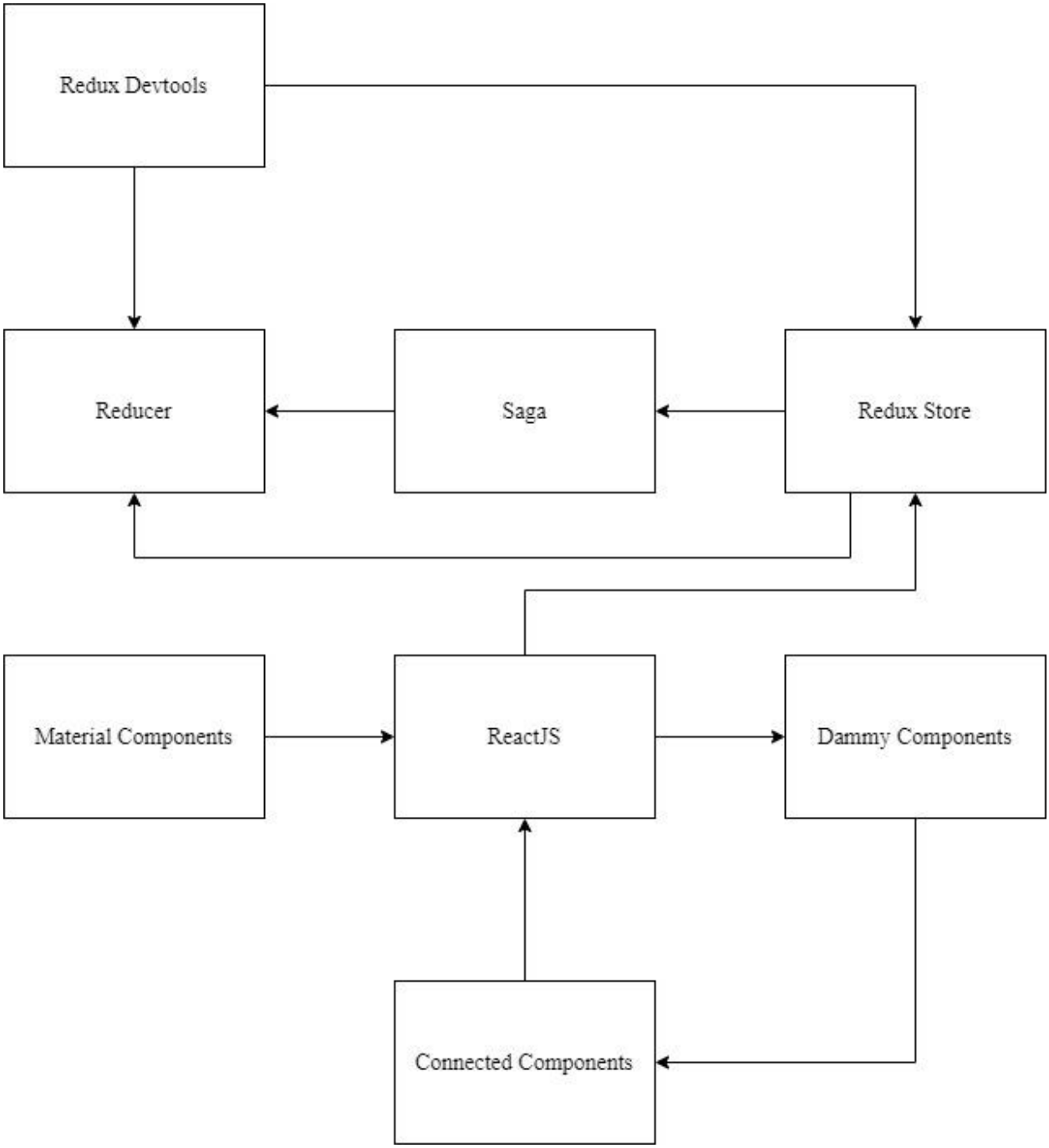
					ДП.6408.05.000 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив					*Система забезпечення якості освітньо-наукової діяльності аспірантури* Пояснювальна записка	Літ. / Аркуш / Аркушів: 1 / 1
Перевірив						
Реценз.						НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІО-64
Н. Контр.		Сімоненко В. П.				
Затвердив						

СТРУКТУРНА СХЕМА

до дипломного проєкту

На тему “Система забезпечення якості освітньо-наукової діяльності аспірантури”

СТРУКТУПНА СХЕМА



					ДП.6408.06.000 Д3		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив					*Система забезпечення якості освітньо-наукової діяльності аспірантури* Пояснювальна записка	Літ.	Аркуш 1 / Аркушів 1
Перевірив							
Реценз.						НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІО-64	
Н. Контр.		Сімоненко В. П.					
Затвердив							